Neural Networks for Basketball Substitution Prediction

Sahil Kumar Karanth
[redacted]
Supervisor: Dr Denes
Subject: Computer Science
[redacted]
Word Count: 3944

Abstract

Substitutions are an integral part of success in basketball and through their prediction, a coach can optimise player rotation. This study approaches the novel problem of predicting the occurrence of substitution plays unlike the majority of literature in this field which focuses on line-up prediction. A play is defined to be a positive or negative moment of possession in the game. Play-by-play data from two National Basketball Association (NBA) teams were used to derive binary features to classify a play, with other match statistics used to derive five additional features: time, quarter number, foul count, time since last score, and score difference. Sequential feature vectors were employed with LSTM neural networks to predict the possibility a substitution would occur ten plays in the future using data from ten previous plays. The final model achieved an accuracy of 60.2% on the test set. The effectiveness of the five additional features was tested, both individually and in all possible combinations by comparing validation accuracy scores. All features improved model accuracy, with the best feature set excluding foul count. The best feature set was used in the hyperparameter tuning process. All pair combinations of six NBA teams were used to train and test the model, resulting in a test accuracy interquartile range of 1.4%. This illustrates the model's ability to generalise to different teams.

1 Introduction

Substitutions occur frequently in basketball and can be a key factor in determining success. Coaches who cycle more players in a match are likely to see fewer errors due to fatigue, while smaller rotations lead to increased game time for crucial players. Using data from the National Collegiate Athletic Association Division I Men's Basketball team, it was found that with larger substitution pools, dynamic plays such as rebounds and steals contributed more to a win [1]. In contrast, smaller rotations emphasised offensive metrics such as shot percentage. Game metric importance was also found to be impacted by the pace of the game [2]. This would likely correlate with the frequency of substitutions.

My research focuses on predicting whether a substitution will occur a set number of plays in the future. This concept could be used to forecast the pace of the game and aid in decision-making.

1.1 Aims and Objectives

This study aims to further research into the use of AI for sports match analysis and explore the novel problem of using basketball play-by-play data to predict the occurrence of substitutions. This concept can improve the level of gameplay by using more optimal player rotations which could increase a team's likelihood of winning. A play is defined as a positive or negative moment of possession in the game.

Historical play lengths are analysed to determine an optimal number of plays used to make a prediction and long short-term memory (LSTM) neural networks are used to make predictions [3]. An ablation study is performed to test the effectiveness of derived features and hyperparameters are tuned to improve model accuracy.

This paper furthers the research across two key areas. Firstly, most related studies aim to predict the starting line-up of players [4] as opposed to the single substitution prediction approach that this paper takes. Secondly, most studies using LSTM networks on basketball data use them for computer vision applications. For instance, these networks have been used to predict 3-point shot success [5] and classify player actions from visual data [6].

2 Background

Basketball is a competitive five-aside sport. The team with the most points at the end of the game wins. Points are scored by getting the ball in the opponent's hoop. To move the ball, a player must bounce it against the ground to "dribble" it. Matches are split into four 12-minute quarters.

2.1 Play Types

Below are definitions of the play types considered in this study:

Layup: A shot taken close to the basket where the ball is released during a jump.

Jump Shot: A shot taken after a vertical jump, where the ball is released into an arc.

Dunk: A shot performed by using one or both hands to put the ball directly into the hoop.

Rebound: A retrieval of the ball after a missed shot bounces off the hoop's rim or backboard. A defensive rebound occurs if the opposing team misses the shot. An offensive rebound occurs if the missed shot was by a teammate or the rebounding player.

Foul: A breach of the game's rules in play or conduct.

Turnover: The act of losing ball possession to the opposing team before shooting the ball into the opponent's hoop.

2.1.2 Substitutions

Any number of substitutions can occur in a match and a player can re-enter the game multiple times. Substitutions cannot take place after a scored point or where "the ball is dead due to a personal foul, technical foul, timeout, infection control or violation" [7].

2.2 Recurrent Neural Networks (RNNs)

Neural networks are directed graphs used in machine learning with neurons that are organised into layers. In feed-forward networks, information is passed through in one direction whereas in RNNs, cyclical connections to nodes in a previous layer allow a neuron's state to recursively impact its future states.

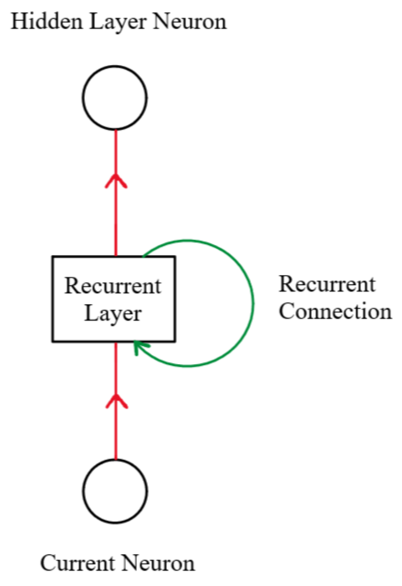


Figure 1: A recurrent neural network connection.

This enables RNNs to process sequential data using their internal state, for instance in time series forecasting or language model applications. However, RNNs suffer from the vanishing or exploding gradient problem which limits their use to small sequence sizes [8].

Weights greater than one in the recursive step grow exponentially as sequence size increases, resulting in oscillating weights during the learning process for large sequence sizes. Weights less than one converge towards zero, greatly increasing training time. Both scenarios restrict the network's capacity to learn. LSTM networks are a variant of the RNN that partially address this issue.

2.3 Long Short-Term Memory (LSTM) Networks

LSTM layers in neural networks consist of memory cells grouped into blocks. At each recursive step, the cell has a cell state, hidden state, and input data. The hidden state represents the network's short-term memory while the cell state is the network's long-term memory. Cells consist of an input, output, and forget gate which determine the contents of the cell state using the cell's current input and previous hidden state. The forget gate removes the previous time step's irrelevant information from the cell state. The input gate is used to update the cell state and the output gate determines the next hidden state. The new cell state and hidden state values are then passed to the next recursive step. For example, a sequence length of ten would repeat this process ten times.

Since the cell state represents the total combined memory of the network, LSTM networks can retain information from more previous steps than traditional RNNs. Hence, they were chosen to allow a greater number of previous plays to be used for substitution prediction. LSTMs address the vanishing gradient problem prevalent in traditional RNNs, but the exploding gradient problem can still occur due to high-magnitude weight updates.

3 Related Work

Related work largely falls into two categories:

1. LSTM networks for computer vision in basketball
2. Models trained with basketball play-by-play data

3.1 LSTMs with Visual Data

LSTMs have been used to predict whether a three-point shot would be successful using the Sport VU visual dataset [5]. The network was trained using three-dimensional coordinates to track the ball at each time step. Additional features were derived such as the ball's distance to the basket's centre and the ball's angle to the basket. LSTMs have also been used with first-person footage to predict a player's position five seconds in the future [9] and to assess a player's performance [10]. Visual data has been used in basketball action classification [6] and shooting classification [11]. These studies analyse complex data and so are unable to comment on trends that can be derived from simple match statistics. Their use of visual data also prevents their work from being applied to leagues without court-side cameras. This study does not encounter these issues.

3.2 Models Using Play-by-Play Data

Logistic regression (LR) and support vector machine (SVM) models have been used to predict streaks in NBA matches [12]. A play was classed as part of a streak if a team scored eight consecutive points. Play-by-play data from the 2014–15 season was used, with 20 plays being used to predict whether a streak would occur in the next 20. A metric using the sum of true positive and true negative classifications was used, where scores greater than 1.0 indicate better than chance. Both models were evaluated on a test set, with the LR model achieving a success rate of 1.647 while the SVM scored 1.319. An ablation study revealed that removing features involving score difference led to the greatest decrease in model performance. This study uses play-by-play data for a largely different problem to substitution prediction. Thus, research into substitution prediction increases understanding of the applications of the data.

Play-by-play data has also been used for line-up optimisation [4]. In contrast, this paper considers single-player changes to further research into team optimisation.

3 Methodology

3.1 Overview

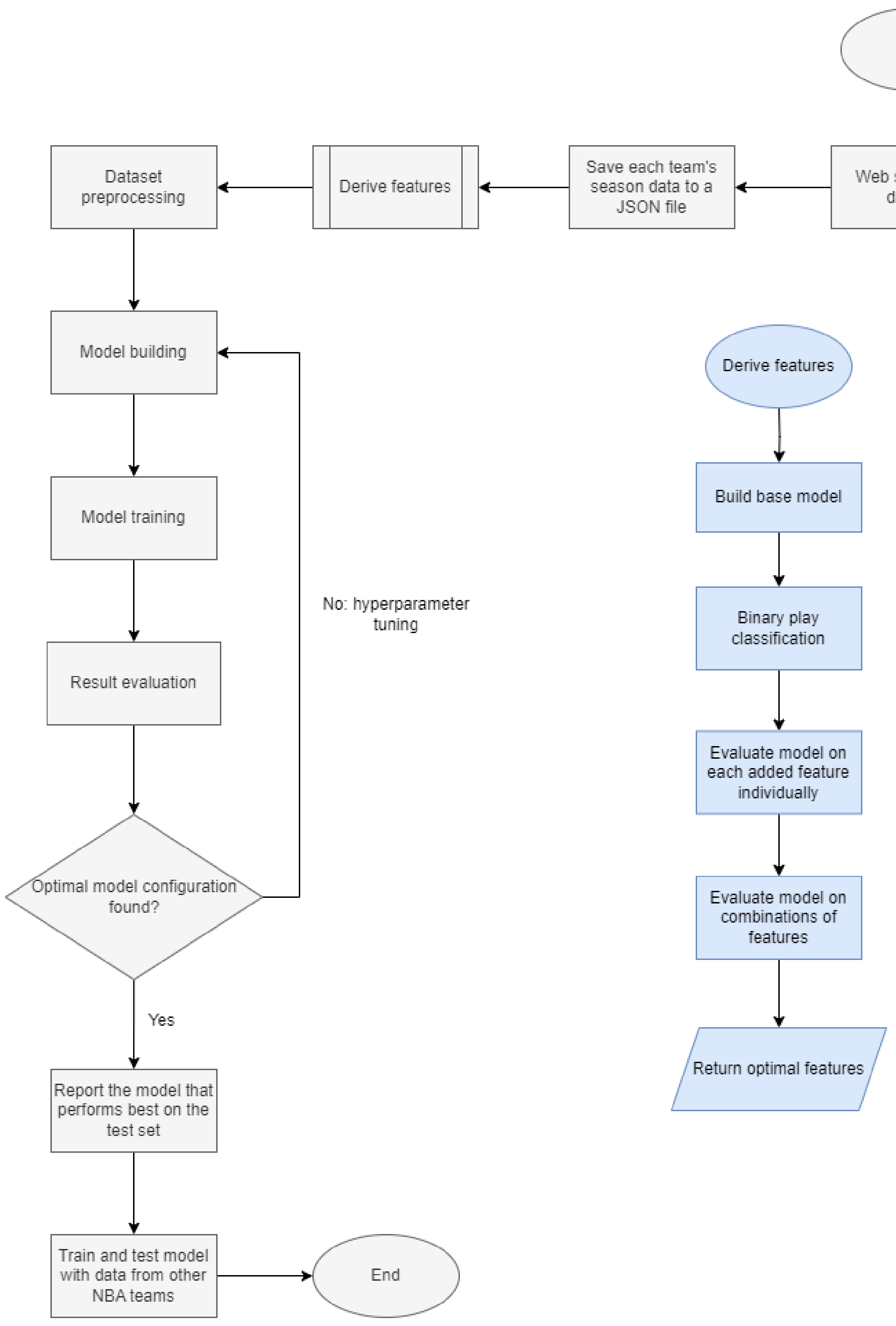
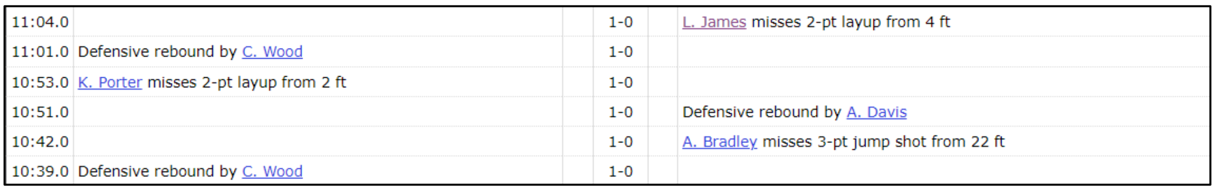

Start
Web scrape the dataset
Save each team's season data to a JSON file
Derive features
Dataset preprocessing
Model building
Model training
Result evaluation
Optimal model configuration found?
No: hyperparameter tuning
Yes
Report the model that performs best on the test set
Train and test model with data from other NBA teams
End
Derive features
Build base model
Binary play classification
Evaluate model on each added feature individually
Evaluate model on combinations of features
Return optimal features

Figure 2: Flowchart illustrating the overall experimental process of this study.

The dataset was web-scraped and used to derive features that classified the play and provided additional contextual information about the moment in the game. Ablation studies were performed to assess the effectiveness of features and model architecture was refined with hyperparameter tuning. The best hyperparameters and set of features were used to train the final model which was evaluated on a test set. Data from other NBA teams were used to train and test the model to assess its ability to generalise to any team.

3.2 Dataset

Game data from the 2015–16 regular season to the 2021–22 regular season for the Milwaukee Bucks and Boston Celtics of the National Basketball Association (NBA) were used in this study. The data was obtained from basketball-reference.com [13] which the majority of related work uses. Data from both teams were treated equally, thus assuming no change in substitution patterns between the teams. The validity of this assumption is tested in Section 4.5. The data is in a tabular format, with plays described in formulaic plain text. Each team's season data was written to a JSON file, with metadata about each game recorded along with each play's team, time, game score, and text description. Python's BeautifulSoup library was used to web-scrape the data.



11:04.0			1-0		L. James misses 2-pt layup from 4 ft
11:01.0	Defensive rebound by C. Wood		1-0		
10:53.0	K. Porter misses 2-pt layup from 2 ft		1-0		
10:51.0			1-0		Defensive rebound by A. Davis
10:42.0			1-0		A. Bradley misses 3-pt jump shot from 22 ft
10:39.0	Defensive rebound by C. Wood		1-0		

Figure 3: Example of tabular play-by-play data from https://www.basketball-reference.com/.

3.2.1 Exploratory Data Analysis

The 2021–22 season for the Milwaukee Bucks was sampled to analyse the lengths of plays. This was done to determine an optimal number of previous plays (n) that would be used to predict the occurrence of a substitution n plays in the future. Increasing n improves the utility of the model by predicting further into the future but can result in a reduction in accuracy due to the vanishing or exploding gradient problem. The lengths of individual plays (Figure 4) and groups of ten plays (Figure 5) were plotted on histograms to analyse their distribution.

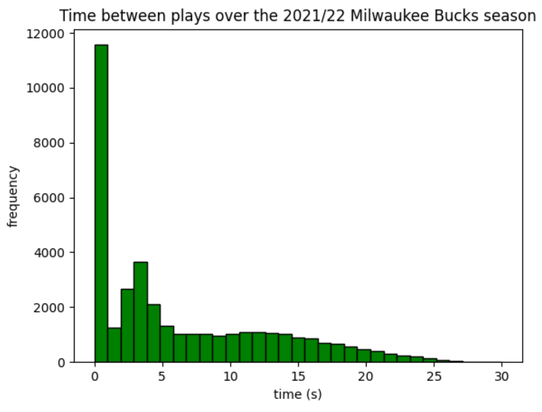


Figure 4: Single-play time duration histogram.

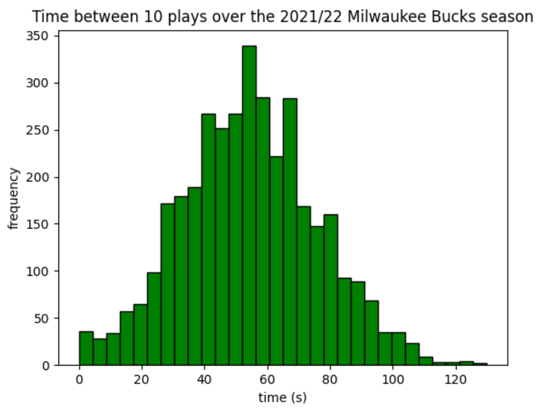


Figure 5: Ten-play time duration histogram.

Figure 4 shows an uneven distribution with a mean single-play time difference of 6.2 seconds. 30.8% of these time differences were in the mode of zero seconds, indicating that a high volume of plays occurred in immediate succession of each other. Figure 5 more closely resembles a normal distribution. This suggests that the mean is a good measure of central tendency in the data and can be used to represent the whole dataset. Thus, a mean of 54.2 seconds is more representative of the data than the single-play distribution and can be used to determine the suitability of an n value of ten. A sequence size of ten was chosen because it balances the utility of predicting further into the future with the shortcomings of LSTM networks over long sequence sizes.

3.3 Feature Derivation

The text description of each play was used to classify it in a binary feature vector. A 1 was assigned to substitution plays and a 0 was assigned to all other plays. Additional aggregate and play definition features were also derived. Feature vectors were labelled with a 1 if in exactly ten plays time a substitution occurred and a 0 for a non-substitution play. The dataset was balanced to have an equal number of substitution and non-substitution resulting training examples. Features are defined from the perspective of Milwaukee or Boston who are referred to as the "target team". For games where these two teams played each other, Milwaukee was chosen to be the target team. Below is a list of the derived features. Features 1-10 were always used whereas features 11-15 were tested for their effectiveness in an ablation study (Section 4.2).

1. Defensive Rebound
2. Offensive Rebound
3. Jump Shot
4. Layup
5. Dunk
6. Foul
7. Turnover
8. Point Value
9. Play Status
10. Team
11. Quarter Number
12. Score Difference
13. Foul Count
14. Time Since Last Score
15. Time

3.3.1 Feature Definitions

Features 1-7 were given a value of 1 if they described the play that occurred and a 0 if they did not.

Point Value: An integer representing the number of available points from a shot. Missed shots are still given a positive value. Non-shooting plays have a point value of 0.

Play Status: A binary feature classifying the play as positive or negative. Missed shots, fouls, and turnovers have a play status of 0 while rebounds and scored shots have a play status of 1. This value is reversed if the play was made by the opposing team.

Team: A binary feature representing which team made the play. The target team is given a team value of 1 while the opposing team is given a team value of 0.

Quarter Number: An integer indicating which quarter the play occurred in from one to four inclusive.

Score Difference: An integer calculated by subtracting the opponent's score from the target team's score. Negative values indicate that the target team is losing while positive values indicate that the target team is winning. 0 represents a draw.

Foul count: An aggregate integer total of the number of fouls in the match up until the current play. The initial value is 0.

Time Since Last score: an integer representing the number of seconds since the target team last scored.

Time: The time left until the end of the quarter in seconds when the play occurred.

3.3.2 Feature Standardisation
Feature standardisation is a process that converts a feature to have a mean of zero and a standard deviation of one. This process prevents higher magnitude features from influencing the neural network's optimisation algorithm more than others. All non-binary features were standardised.

3.3.3 k-Fold Cross-Validation
k-fold cross-validation involves subdividing a dataset into k subsets (folds), training a model on $k-1$ sets, and evaluating it on the final set. This method greatly increases training time but reduces overfitting and results in a more conclusive understanding of model performance. k-fold cross-validation was used when hyperparameter tuning.

3.4 Model Creation
TensorFlow's Keras API was used to create the model. An initial LSTM model structure which is consistent with the literature was chosen for feature comparison tests. This structure is described in Table 1. Informal experimentation led to activation function choices.

Table 1: Initial model architecture summary

Layer	Neuron Count	Activation Function
LSTM	64	Tanh
LSTM	64	Tanh
Dense	32	ReLU
Dense	1	Logistic Sigmoid

3.4.1 LSTM Layers
Each LSTM layer consisted of 64 neurons and used the tanh (hyperbolic tangent) activation function. The tanh function converts inputs to a value between -1 and 1. Ridge regression was applied to LSTM layers. This combats overfitting by adding a penalty term to the loss function equal to the sum of the squares of the weights.

3.4.2 Dropout and Batch Normalisation Layers
Dropout and batch normalisation layers were used after each LSTM layer to combat overfitting. A dropout was also added after the first dense layer. Dropout layers randomly exclude neurons from the training process of an epoch. As no neurons are excluded from validation data, validation accuracy is initially higher than training accuracy. All dropout layers in the model used a dropout rate of 20% which is consistent with the literature. Batch normalisation scales the output of an activation function, similar to feature standardisation.

3.4.3 Dense Layers
Neurons in dense layers have connections to every neuron in the previous layer. The model's first dense layer uses 32 neurons and a rectified linear unit (ReLU) activation function which outputs 0 for negative values but keeps the value unchanged if it is positive.

The output dense layer contains one neuron and uses a logistic sigmoid activation function which converts values to an output between 0 and 1. Outputs greater than 0.5 are classed as a substitution while outputs less than 0.5 are classed as no substitution. Values closer to 0 or 1 indicate more confident predictions.

3.4.4 Model Compilation

During compilation, the model's loss function, learning rate, optimiser, and optimisation metrics are defined. The learning rate was set to 1.0×10^{-4} along with a batch size of 32. The accuracy metric and Adam optimiser were used. Binary cross entropy loss was used and is given by,

$$Binary\ Cross\ Entropy\ Loss = \frac{1}{N}\sum_{i=1}^{N} -(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

where y_i is the ground truth value of class 1, $\hat{y}_i$ is the predicted probability of class 1, and N is the batch size.

4 Experimental Results and Discussion

In this section, the final model's results are first presented and discussed. Feature ablation study and hyperparameter tuning results are then described and optimal hyperparameters and features are reported. Data from paired team combinations are tested to assess the model's ability to generalise.

4.1 Final Model

The final model was trained using 80% of the dataset over 70 epochs and tested on 20%. The best derived features and hyperparameters were used as per the feature ablation study and hyperparameter tuning results. The baseline for this study was 50% as this would be the accuracy for substitution prediction using random guesses. An accuracy of 60.2% was achieved on the test set, meaning the model performed better than the baseline. The model was trained on balanced data, meaning substitution and no substitution mispredictions were penalised equally. This assumes that the probability of any play being a substitution is 50%. However, over the 2015–16 to 2021–22 seasons for the Milwaukee Bucks and Boston Celtics, the probability of a play being a substitution was found to be 10.4%. Thus, the model's test set accuracy is not a good estimate of how it would perform in a real match scenario.

Line-up outcome prediction has been shown to reach 68% accuracy with graph theory [14]. However, it is difficult to directly compare this percentage or other figures in literature with this study's results as no other papers attempt to predict the occurrence of substitutions.

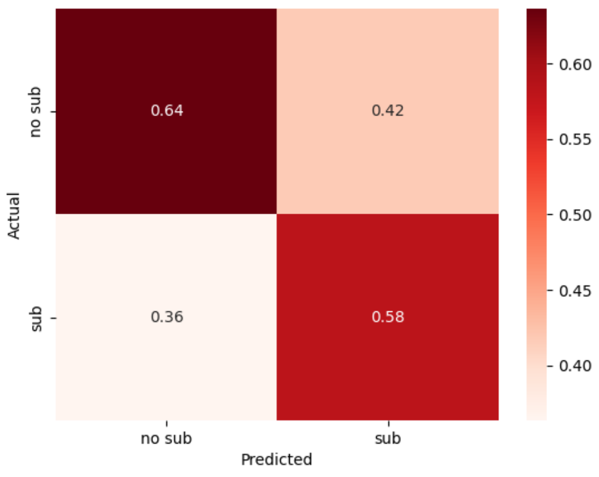


Figure 6: Final model's normalised confusion matrix. Quadrant values represent the percentage of test set predictions that belong to the category.

Figure 6 illustrates that the model is more accurate when it predicts no substitution occurring and less accurate when it predicts a substitution.

4.2 Feature Ablation Study

The additional features were added and tested individually with the model architecture described in Section 3.4 over 100 epochs. 80% of the data was used for training, 10% for validation, and 10% for testing.

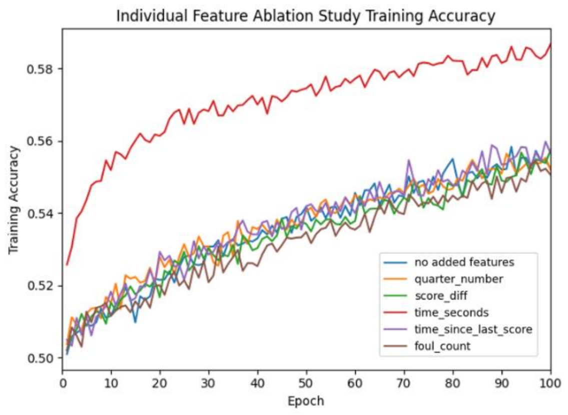


Figure 7: Training accuracy–epoch number graph for individual feature tests.

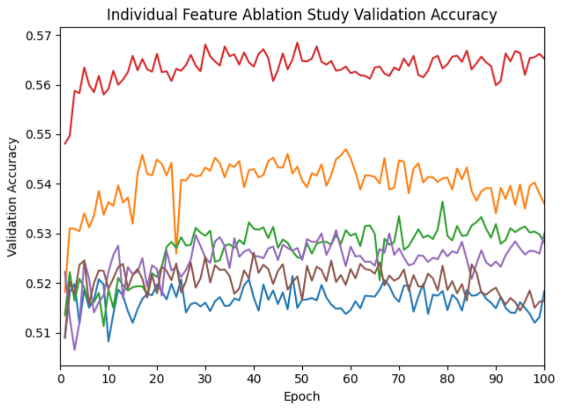


Figure 8: Validation accuracy-epoch number graph for individual feature tests.

Time led to the greatest increase in model validation accuracy, followed by quarter number. This indicates that when a play occurs is crucial for substitution prediction. All five tested features were considered for the combination ablation study (Section 4.2.1) as they all improved overall model validation accuracy.

4.2.1 Feature Combination Ablation Study

An additional 26 models were trained with the base architecture, accounting for all possible combinations of the five investigated features. This was done to determine the best combination of these features as it is difficult to predict the effectiveness of a set of features in neural networks without experimentation. Each model's training was stopped early if its validation accuracy did not improve after 20 epochs to reduce overfitting and training time. The highest-performing model included all additional features except for foul count. It ran for 64 epochs and had an average validation accuracy over the final five epochs of 60.3%. While this was only 0.9% greater than the second-best model, it took 29 fewer epochs to train. Thus it was deemed to be a sizeable improvement from the second-best model and its feature set was used for hyperparameter tuning, the final model, and team combination testing.

4.3 Hyperparameter Tuning

Hyperparameters specify characteristics of the training process and the model's architecture, influencing model performance. Common neural network hyperparameters include neuron count, batch size, learning rate, and the number of layers. Tuning with a grid search involves training a model with every combination of specified hyperparameters. However, this is too computationally demanding for this application as too many models would need to be trained. Instead, a random search was chosen which involves testing n random parameter configurations, allowing the extent of the search to be altered depending on time and resource availability. Scikit-learn's *RandomizedSearchCV* function was used [15] along with the scikeras package's *KerasClassifier* wrapper. 5-fold cross-validation was used with 60 epochs per fold and 20 hyperparameter configurations were trained, totalling 100 fits. Table 2 summarises the hyperparameters tuned, and the best model's hyperparameter values.

Table 2: Hyperparameters tuned, their possible values, and the best values obtained from the search.

Hyperparameter	Possible Values	Best Value
Number of LSTM layers	2, 3	2
Number of Dense layers	1, 2, 3	2
LSTM layer neuron count	32, 64, 128	64
Dense layer neuron count	32, 64, 128	32
Learning rate	$(1 \times 10^{-5}), (1 \times 10^{-4})$	(1×10^{-4})

The hyperparameters specified in the "Best Value" column of Table 2 were used for team combination testing (Section 4.5) and the final model (Section 4.1).

4.5 Team Combination Testing

In all previous sections, data from the Milwaukee Bucks and Boston Celtics from the 2015–16 regular season to the 2021–22 regular season were used. In this section, data from paired combinations of six NBA teams were used to train a model and assess the model's ability to generalise with any team. The following teams were sampled as part of the test:

1. Milwaukee Bucks
2. Boston Celtics
3. Miami Heat
4. Dallas Mavericks
5. Denver Nuggets
6. Utah Jazz

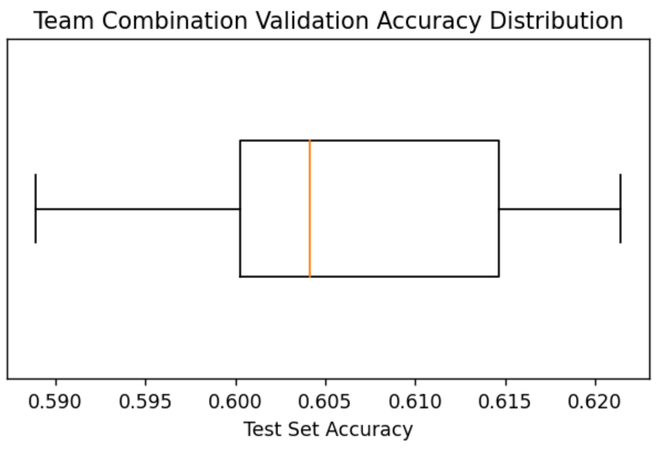


Figure 9: Boxplot displaying team combination accuracy distribution.

Figure 9 illustrates the distribution of test accuracy between all team combinations. For each combination, 80% of the data was used for training and 20% was used for testing. The median accuracy was 60.4% with an interquartile range of 1.4%. This small variation in team combination accuracy and the fact that all models performed better than the 50% baseline conveys the model's ability to generalise well with any team. However, the range of 3.3% and negatively skewed data convey the model's limitations in performing optimally with every team.

5 Limitations of the Model

As mentioned in Section 4.1, the model's utility in a real match scenario is limited due to being trained on balanced data. Future work could attempt to train the model on a stratified dataset that is representative of substitution probabilities. Many other factors influence substitutions in basketball in addition to match and play-by-play statistics. Factors such as injuries, fatigue, and a coach's substitution patterns were not considered in this study and could likely have improved model performance. However, some factors such as coach favouritism or racial bias could be challenging to include. Computational constraints prevented a full-scale hyperparameter grid search and reduced the number of iterations possible with a random search. It is likely that with more resources, superior hyperparameters could be found. Additional teams could also have been used in combination tests given more resources, enabling more accurate substitution similarity conclusions to be drawn.

6 Conclusion

This study has shown that it is possible to predict the occurrence of substitutions using play-by-play data, meaning that with more data, it is likely that the substitution of specific players can be predicted. LSTM neural networks were used with sequential feature vectors derived from play-by-play data and optimal features and hyperparameters were tested for. Fundamental match statistics such as quarter number and time led to the greatest increase in model performance and the best feature set was found to include all derived features except foul count. The final model achieved an accuracy of 60.2 %. Data from all paired combinations of six NBA teams were used to train models, resulting in test accuracies that had an interquartile range of 1.4 %. This illustrates the model's capability to generalise well across different teams. Future work could incorporate player-specific statistics such as position or a composite skill metric. A more extensive hyperparameter search could also be performed.

Acknowledgements

I would like to thank Dr Denes for his constant support and guidance throughout this project.

Source Code

This study's code is available at https://github.com/skkaranth1/Neural-Networks-for-Basketball-Substitution-Prediction

Bibliography

[1] D. C. Clay and K. E. Clay, "Player rotation, on-court performance and game outcomes in NCAA men's basketball," *International Journal of Performance Analysis in Sport,* vol. 14, no. 2, pp. 606-619, 2014.

[2] G. Csataljay, M. Hughes, N. James and H. Dancs, "Pace as an influencing factor in basketball," *Research methods and performance analysis,* p. 178, 2011.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[4] L. Grassetti, R. Bellio, G. Fonseca and P. Vidoni, "Estimation of lineup efficiency effects in Basketball using play-by-play data," *Book of Short Papers SIS2019,* pp. 363-370, 2019.

[5] R. C. Shah and R. Romijnders, "Applying Deep Learning to Basketball Trajectories," *arXiv preprint arXiv:1608.03793,* 2016.

[6] C. Ma, J. Fan, J. Yao and T. Zhang, "NPU RGB+ D Dataset and a Feature-Enhanced LSTM-DGCN Method for Action Recognition of Basketball Players," *Applied Sciences,* vol. 11, no. 10, p. 4426, 2021.

[7] NBA, "RULE NO 3: Players, Substitutes and Coaches," NBA, 2020. [Online]. Available: https://official.nba.com/rule-no-3-players-substitutes-and-coaches/. [Accessed March 2023].

[8] Y. Bengio, P. Frasconi and P. Simard, “The problem of learning long-term dependencies in recurrent networks,” *IEEE international conference on neural networks,* pp. 1183-1188, 1993.

[9] S. Su, J. P. Hong and J. Shi, “Predicting Behaviors of Basketball Players from First Person Videos,” *Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 1501-1510, 2017.

[10] G. Bertasius, H. S. Park, S. X. Yu and J. Shi, “Am I a baller? basketball performance assessment from first-person videos,” *Proceedings of the IEEE international conference on computer vision,* pp. 2177-2185, 2017.

[11] H. Gammulle, S. Denman, . S. Sridharan and . C. Fookes, “Two stream lstm: A deep fusion framework for human action recognition,” *IEEE winter conference on applications of computer vision (WACV),* pp. 177-186, 2017.

[12] W. LaRow, B. Mittl and V. Singh, “Predicting Momentum Shifts in NBA Games,” 2014.

[13] “Basketball-Reference.com - Basketball Statistics and History,” Sports Reference LLC, [Online]. Available: https://www.basketball-reference.com/. [Accessed February 2023].

[14] M. Ahmadalinezhad, M. Makrehchi and N. Seward, “Basketball lineup performance prediction using network analysis,” *Proceedings of the 2019 IEEE/ACM international conference on advances in social networks analysis and mining,* pp. 519-524, 2019.

[15] F. Pedregosa, G. Varoquaux , A. Gramfort , V. Michel , B. Thirion , O. Grisel , M. Blondel , P. Prettenhofer , R. Weiss , V. Dubourg and J. Vanderplas , “Scikit-learn: Machine Learning in Python,” *the Journal of machine Learning research,* vol. 12, pp. 2825-2830, 2011.