Q-Ball: Modeling Basketball Games Using Deep Reinforcement Learning

Chen Yanai[1], **Adir Solomon**[1], **Gilad Katz**[1], **Bracha Shapira**[1], **Lior Rokach**[1]

[1]Ben-Gurion University of the Negev, Beer-Sheva, Israel
{chenyan,adirsolo,liorrk}@post.bgu.ac.il, {giladkz,bshapira}@bgu.ac.il

Abstract

Basketball is one of the most popular types of sports in the world. Recent technological developments have made it possible to collect large amounts of data on the game, analyze it, and discover new insights. We propose a novel approach for modeling basketball games using deep reinforcement learning. By analyzing multiple aspects of both the players and the game, we are able to model the latent connections among players' movements, actions, and performance, into a single measure – the *Q-Ball*. Using *Q-Ball*, we are able to assign scores to the performance of both players and whole teams. Our approach has multiple practical applications, including evaluating and improving players' game decisions and producing tactical recommendations. We train and evaluate our approach on a large dataset of National Basketball Association games, and show that the *Q-Ball* is capable of accurately assessing the performance of players and teams. Furthermore, we show that *Q-Ball* is highly effective in recommending alternatives to players' actions.

Introduction

Basketball is one of the most popular sports around the world. Its wide popularity has contributed to the growth of the US National Basketball Association (NBA) (Manner 2016). The NBA is the most important and influential professional basketball league in the world (Manner 2016).The rise of the game's popularity sparked rapid growth both in data collection and the development of advanced methods for evaluating different aspects of the game (Torres 2013; Macdonald 2020; Sampaio et al. 2015).

One of the most important and innovative developments in sports-related data collection is the SportVU player tracking system[1] [2]. SportVU records all player positions on the court, as well as the ball's, and it does so with a very high resolution of 25 frames per second. At the beginning of the 2013-14 NBA season, every team in the league installed the SportVU system on its court. Analyzing this type of data enables us to evaluate players' movements and game decisions (Sicilia, Pelechrinis, and Goldsberry 2019), and providing different tools for improving their game.

While previous studies (Neiman and Loewenstein 2011; Wang et al. 2018) did use deep reinforcement learning (DRL) on basketball games, their goals were limited to modeling a specific and relatively small set of player actions. In this study we propose a DRL-based framework for modeling full basketball games. As a result, we were able to develop the *Q-Ball* measure, which quantifies the impact of actions by specific players on a given game. Using *Q-Ball* could be beneficial for several common scenarios; during a recess, the coach can "replay" recent states and present alternatives should they happen again. In addition, when considering tactics for the remainder of the game (during timeouts, quarters intermissions, and halftime break), the coach can explore alternative moves. *Q-Ball* could also be beneficial when changing players, the coach can better evaluate which player will be most effective on average in the current lineup. Moreover, using *Q-Ball* offline can allow scouts and managers to identify potential recruits with promising decision-making skills.

Our novel approach enables us to jointly model not only the discrete actions players make during the game (e.g., pass, shot), but also continuous aspects such as player velocity. Moreover, our framework relies on the rich contextual information such as players' attributes (e.g., height and weight), players and teams identifiers, the quarter and shot clocks, etc. Thus, the *Q-Ball* measure is able to capture and model latent connections among multiple contextual factors and their discrete and continues actions. These capabilities enable us to accurately model the quality of various aspects of the game, from a single action to the performance of an entire team.

We evaluate our approach on a dataset comprised of 619 real NBA games in the 2015-16 season, derived from the SportVU system and play-by-play, which describes the events that occur during a game, e.g., shots, steals, and rebounds (Vračar, Štrumbelj, and Kononenko 2016). Merging the data derived from the SportVU system and the play-by-play results in a high-resolution information dataset that captures both players' movements and events. This study's main contributions are as follows:

- We use a novel method that enables us to combines the discrete actions and continuous actions of basketball

[1]http://grantland.com/features/the-toronto-raptors-sportvu-cameras-nba-analytical-revolution

[2]https://www.espn.com/blog/playbook/tech/post/_/id/492/492

players into a single measure (*Q-Ball*). This representation enables us to detect hidden patterns and dependencies in the game.

- We propose a novel method for evaluating and improving player's decision-making. We achieve this by comparing the *Q-Ball* values of a player's actions to those of alternative simulated actions.
- We interpret the *Q-Ball* values using the game theory-based framework of SHAP (SHapley Additive exPlanations) (Lundberg and Lee 2017). Using SHAP allows us not only to present and analyze the "what-if" scenarios (i.e., assess the outcomes of different decisions than the one taken), but also answer the "why" (i.e., why the player should have done different action).
- We utilize the *Q-Ball* impact measure, which evaluates the effect of a player on his team's offensive capabilities. This measure can detect players that have a positive impact on the offense, which is often difficult to detect using traditional statistics. We also show that the *Q-Ball* correlates with NBA teams' scoring capabilities and All-Star players, who are considered the best players in the NBA.

Related Work

Recent studies (Liu and Schulte 2018; Liu et al. 2020; Liu, Zhu, and Schulte 2018; Wang et al. 2018; Decroos et al. 2019; Wang et al. 2020) applied DRL for the analysis of various fields of sport; in the work of (Liu and Schulte 2018), the authors modeled ice hockey games in order to evaluate player performance. The authors used a DRL-based solution to learn the Q-values (i.e., the "quality" of different actions) with respect to the game's context, e.g., puck's location coordinates, game time remaining. To this end, the authors applied a LSTM-based architecture. Later in this study, the Q-values framework was extended to the team level. The study provides strong evidence that DRL-based algorithms can accurately assess the quality of individual actions by players, as well as player/team quality overall.

Several recent works used DRL for modeling basketball games; Seid et al. (Seidl et al. 2018) proposed combining the extracted context of basketball games together with a deep limitation learning algorithm for the ghosting of basketball players. The authors demonstrated their method's abilities by sketching players' movements and actions. While this work was proven effective in the task of ghosting basketball players, it did not created a new measure for assessing players' performance in real time. Moreover, deep limitation learning based methods have several limitations (Kostrikov et al. 2018); (1) they are not optimal due to the implicit bias in the reward function; (2) they require expert demonstrations, and; (3) they require great number of interactions with the environment to successfully imitate the experts from which they learn. For this reason, more efficient DRL-based algorithms such as actor-critic (which is also used in this study) are also explored in this context (Kostrikov et al. 2018).

In recent work (Wang et al. 2018), the authors used a DRL to model the decision of whether or not to use a double team (i.e., assigning two players to guarding one player from the opposing team) during a basketball game. Their DRL model was based on the study of (Van Hasselt, Guez, and Silver 2016), which used two value functions that were learned based on two sets of weights. Their results could then be applied for recommending which players should be double-teamed, and who could be left unguarded.

All of these methods use a relatively simplistic approach that only model specific parts of the game (e.g., the possession end results). Thus, these methods are not comparable with our suggested approach, in which we model the game in its entirety. Furthermore, we manage to provide a unique, realistic, and suitable measure, *Q-Ball*, for accurately modeling basketball games. This is the case for the following reasons: (1) we use an extension of the DDPG algorithm, which enables us to captures the dependencies between discrete and continuous actions; (2) we use rich contextual information, such as players' features and movements; (3) instead of focusing on a specific and limited set of actions (Wang et al. 2018; Sicilia, Pelechrinis, and Goldsberry 2019), we are able to represent all possible states and actions in all of the basketball games in our dataset, and; (4) we show that *Q-Ball* has a high correlation with the performance of actual players and teams and can be beneficial for improving the decision-making of both players and coaches.

The Proposed Method

States, Actions & Rewards

Our DRL agent always takes the perspective of the team which is on the offense, while modeling the defending team's actions as part of its state representation. We chose to focus on one of the two perspectives of the game as it simplifies the training process and makes our model more explainable. It should also be noted that our model can be just as easily applied to defense. Moreover, since the two teams repeatedly switch sides, training two models is the logical course for a real-world team.

States. Our state space consists of all possible combinations of the players/ball positions on the basketball court. We augment this representation with additional information regarding the elapsed time and player statistics. More formally, we define the state at time t as $s_t = \{M_t, F_t\}$, where $M_t = \{(p1_x, p1_y), (p2_x, p2_y), .., (ball_x, ball_y), ball_p, s_{clock}, g_{clock}\}$. The variables (pi_x, pi_y) represent the coordinates of players along the x and y axes of basketball court i, and $(ball_x, ball_y)$ similarly represent the coordinates of the ball. The variable $ball_p$ denotes the index of the player in possession of the ball, and s_{clock} and g_{clock} represent the remaining possession time and game time, respectively. The vector F_t consists of the following static information: participating teams' IDs, participating players' IDs, and the height, weight, role, and shooting percentage of each participating player.

Actions. Each of our actions consists *both of a discrete and a continuous component*. We define action $a = \{AD, AC\}$, where AD is a discrete variable denoting the action type and

AC is a set of continuous variables describing the characteristics of the action.

- AD – the possible discrete actions are as follows: jump shot, dunk, layup, hook shot, and the five possible passes (one cannot pass the ball to oneself, but this setting was used for maintaining the indices of the players static throughout the game).
- AC – the velocities of all the players of the agent's group, as well as that of the ball. Therefore, the vector for a given time t is as follows: $AC_t = \{(v_{p1_x}, v_{p1_y}), (v_{p2_x}, v_{p2_y}), .., (v_{ball_x}, v_{ball_y}, v_{ball_z})\}$. Please note that the velocities of the players are two-dimensional while the ball's is three-dimensional.

It is important to note that our continuous actions are generated *for the entire team*. Each player in the analyzed team is assigned with their own individual set of continuous values, which determine the optimal action (according to our model's assessment) that the player should carry out.

In our experiments we distinguish between two set of actions: *actual actions* $a = \{AD, AC\}$ which took place in reality, and *suggested actions* $\hat{a} = \{\hat{AD}, \hat{AC}\}$ which are alternatives proposed by our model. We elaborate on this further in the following section.

The Reward Function. We assign a reward for every time step in the game. The reward for each time step t, denoted by r_t, reflects the actual scores assigned by the rules of the game (i.e., as close a representation of reality as possible). The reward at time step t receives one of three values: *a)* a value of $\{+2, +3\}$ if a successful shot has been made (the value is based on the type of the shot); *b)* A value of -0.5 if the opposing team takes hold of the ball, and; *c)* 0 otherwise.

The negative reward of -0.5 was empirically set, and its goal was to actively encourage our DRL agent to avoid scenarios where the ball is dropped. We use a value of -0.5 as higher negative values could imply that losing the ball inevitably results in points for the opposing team, which is not always the case.

Model Architecture

Our goal is to evaluate the performance, i.e., "quality of play", of individual players and whole teams. We therefore use an actor-critic architecture (Konda and Tsitsiklis 2000), which outputs a value for each possible state-action combination. Actor-critic architectures consist of two main components: *a)* the *actor*, whose task is to improve the policy (i.e., strategy) executed by the architecture, and; *b)* the *critic*, whose task is to assess the current policy and provide feedback to the actor. We use the assessments made by the critic for our final assessment of game quality.

An overview of our proposed architecture is presented in Figure 1. We implement the DDPG algorithm (Lillicrap et al. 2015), which is a popular version of the actor-critic architecture. More specifically, we build upon the implementation presented in (Hausknecht and Stone 2015), which is an expansion of the original algorithm. This extension includes two important additions: first, we use a replay buffer (Lillicrap et al. 2015; Zhang and Sutton 2017), which stores

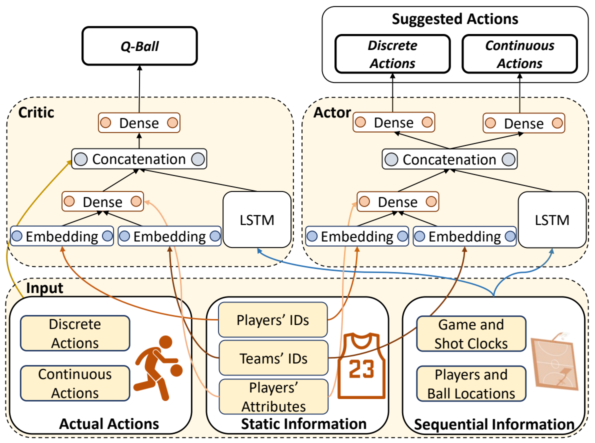

Figure 1: Overview of our proposed DRL model for outputting the *Q-Ball* measure.

previously-analyzed games. The buffer routinely submits these games as input to the architecture in order to prevent the architecture from "forgetting" previously-analyzed cases. Secondly, we use "soft" target updates (Lillicrap et al. 2015), which were shown to improve the architecture's performance. We now describe the various components of our architecture in detail.

The input. Similarly to our state representation, our input has both a static and a sequential (dynamic) component. The static component is identical to the F_t vector, and is used to describe the properties of the participating players and their teams. The sequential component consists of the M_t vector of the current time step t, concatenated to the previous 15 M_t vectors – $\{M_{t-15}, .., M_{t-1}\}$. These vectors are ordered chronologically, and are provided as input (one time step at a time) to the LSTM networks of both the actor and the critic.

The Actor. The actor receives as input both the static and the sequential inputs. For the static inputs, we create separate embedding representations for the players and teams IDs. The players' attributes (e.g., height) are fed directly to the subsequent dense layer (since the latter is numeric, no embedding was deemed necessary). The sequential input is fed to an LSTM architecture (Hochreiter and Schmidhuber 1997), and the final output of the network is then concatenated to that of the dense layer representing the static input.

Our actions contain both a discrete component (the type of the action) and a continuous one (e.g., the velocity of each player and ball throw angle). To represent this duality, our actor architecture has two outputs, one for each component. The discrete actions output uses the softmax function to select a discrete action, while the continuous actions output uses the $tanh$ function. The final output of the actor is therefore not only *what* to do, but *how* to do it—not only where to run, but at what speed, for example. In this regard our approach provides a level of detail not possible in previous studies.

The Critic. The critic receives (and models) the same input as the actor, but with the addition of the *actual actions* that occurred during the game. These actual actions are added during the concatenation phase that combines the discrete and continuous information, exactly as in the actor module. The critic then proceeds to assign a score to the actual actions, while taking into consideration the static and sequential information. We denote this score as *Q-Ball*. This measure is the main output of our approach, and later in this study we demonstrate how it can be used to rank players and teams. We also use *Q-Ball* to rank the alternative actions to those actually taken by replacing the actual actions with the suggested actions (i.e., the output of the actor). Note that a higher *Q-Ball* for the actions of a player means that the model considers the combined static/sequential characteristics of the action effective. This means that the right type of action (e.g., pass the ball), if executed poorly (e.g., moving to left instead of the right), will result in a low *Q-Ball* score.

The critic network is updated using the temporal difference (TD) update, as shown in (Watkins and Dayan 1992):

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a_{t+1}} Q'(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \quad (1)$$

where Q represents the critic network, γ is the discount factor for future rewards, and α is the learning rate.

By transforming the temporal difference update equation to the critic's loss function and adjusting it to the continuous action space, we aim to minimize the following loss function:

$$L_Q(s_t, a_t|\theta) = (Q(s_t, a_t|\theta^Q) - (r_t + \gamma Q(s_{t+1}, \mu(s_{t+1}|\theta^\mu)'|\theta^Q)))^2 \quad (2)$$

Where μ is the actor network, θ^μ are the network's parameters, and θ^Q are the critic's network parameters.

While our proposed architecture builds upon the one presented in (Hausknecht and Stone 2015), we wish to point out an important difference. While the output of (Hausknecht and Stone 2015) is a specific discrete action and its continuous parameters, our proposed approach is used to simultaneously assign actions to multiple separate entities, i.e., an entire basketball team.

Data Collection and Preprocessing

Data Collection. The technological innovations of recent years now enable extremely accurate modeling of the game. As a result, we are able to create a rich and accurate representation and use it as input for our DRL agent. In this study, we used (and combined information from) the following two datasets:

- **The moments dataset** – This dataset was collected by the SportVU tracking system[3]. The data is partitioned to games, where each game is a sequence of events. An event contains players' IDs, teams' IDs, and a sequence of moments. A moment is composed of the players' coordinates (x, y), ball coordinates (x, y, and z), and the game and shot clocks. The dataset contains data of 619 games from the 2015-2016 NBA season.
- **The play-by-play dataset**[4] – This dataset describes every play results that occurred during the game, such as shots, rebounds, fouls, and substitutions. We extracted the play-by-play data for all games in the moments dataset. For our purposes, we selected the following play results types: shots, rebounds, fouls, and turnovers. We also extracted the play result's textual description, relevant players, game clock, and ID.

[3]https://www.statsperform.com/team-performance/football/optical-tracking/

Preprocessing. We preprocessed the dataset in order to create episodes from the *game possessions*, with possession being the period of time (up to 24 seconds) in which one team was in control of the ball. To create a possessions dataset annotated with the outcome of each possession (e.g., score, losing the ball), we merged the datasets mentioned above using the game and shot clocks. Based on the ball locations, i.e., coordinates, we extracted actions (passes, shots etc.) and the identity of the current player handling the ball. Based on the event description, we determined whether a shot is one of the following types: jump shot, layup, dunk, or hook shot. Additionally, we used the players' locations to compute their velocities in each axis for each moment; the velocity delta between two moments is regarded as a player's movement actions.

Episodes that include shots end when the player releases the ball, allowing us to assign the reward to the shooting action. In contrast to most works in the DRL field, we do not use an environment for recording episodes; our dataset serves as a recorded replay of episodes. The episode is defined as a sequence of states, actions, and rewards.

Evaluation

Experimental Setup

Data Sampling. We sample the episodes dataset every 0.24 seconds in order to: (1) reduce the amount of data and avoid the exploding gradient problem (Bengio, Simard, and Frasconi 1994), and; (2) improve the *Q-Ball* by adjusting the actions' change rate to the speeds of real players (so each sample reflects some change in the state of the game). The state's sequential data contains the data from 16 such moments (i.e., consecutive samples), and each state accumulates the data from the last four seconds. We also filter episodes with lengths of less than four seconds; as a result, a single state contains four seconds worth of data.

Model Parameters. We use the grid search strategy to optimize the model's parameter. The final setting used in our experiments is as follows:

- In order to evaluate aggregated measures, such as overall offensive success rate of a given team, we use a discount factor value of 0.8. To evaluate individual measures, such as shooting players' efficiency, we use a discount factor value of 0.95. The higher discount factor for individual players provides an incentive for our model to place greater emphasis on the actions of individual players.

[4]https://www.basketball-reference.com/

- We use stochastic gradient descent (SGD) as the optimizer, and set the replay buffer size at 50,000. We also set the batch size at 32, which determines the number of transitions sampled from the replay buffer.
- We use two different learning rates: the actor's learning rate is 0.0001, and the critic's learning rate is 0.0002.
- We set each of the embedding layers at 10 neurons in size, and each of the internal dense layers at 64 neurons in size. These neurons are with the hyperbolic tangent ($Tanh$) activation function; the LSTM has 50 cells. The final dense layer in the critic is initialized with the linear activation function with a single neuron, which outputs the *Q-Ball* values. In the actor, to assess the suggested discrete actions in the final dense layer, we set the number of neurons to 11. This value is equal to the number of possible discrete actions, and we initialize it with the $softmax$ activation function. To assess the continuous actions we set the number of neurons to 13 which is equivalent to the number of possible continuous actions, all with the $Tanh$ activation function.

Hardware We implement our DRL model on a machine with the following settings: a GPU card of RTX 2080, CPU of Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz and 72G of RAM, Samsung DDR4 2666 MHz.

Evaluating a Player's Impact on a Lineup

The goal of this analysis is to assess *Q-Ball*'s ability to predict the contribution of any given player to a group's lineup. In other words, our aim is to predict whether the addition or replacement of a player will improve the team's offensive capabilities, measured by the overall number of points in a game/season.

Before addressing the inherently difficult problem of predicting a player's contribution to his team's offensive capabilities, we perform a preliminary analysis. We calculate the average *Q-Ball* for all the unique lineups that played together for at least 50 episodes throughout our data. The top five lineups are presented in Table 1. Examining the results, we observe that a third of the top 15 lineups belong to the Golden State Warriors, which broke the NBA record for the most wins in a regular season during the analyzed season.

Next, we turn our attention to predicting individual players' contribution to the lineup. It should be noted that measuring the overall impact of a player on the success of the offense is inherently difficult, as a player can impact the offense without performing any significant actions. For example, when a good three-point shooter is standing behind the arc, he forces his defender to stay within reach and therefore prevents him from helping other defenders. Replacing this player with a bad three-point shooter allows the defender to be more aggressive in helping his teammates with their defensive assignments, reducing the offense chances to score.

In this experiment, we define the player's *Q-Ball* impact on a specific lineup as the difference between the lineup's average *Q-Ball* value when the player was part of the lineup to the average *Q-Ball* when he was not. The *Q-Ball* impact measure allows us to evaluate a player's overall effect on the success of his team's offense. The *Q-Ball* impact creates a

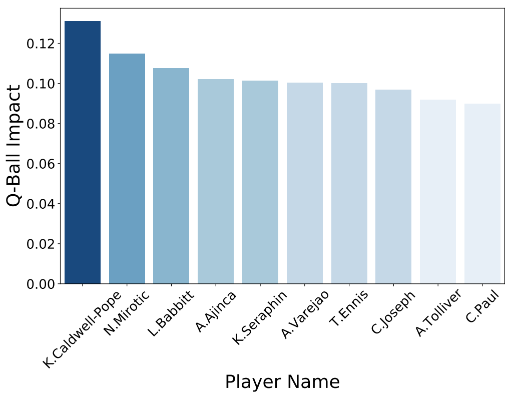

Figure 2: Top 10 players with the highest *Q-Ball* impact values.

more accurate evaluation of a player's offensive effect, since it considers all of the other players playing with him. Figure 2 presents the top 10 players with the best overall *Q-Ball* impact values. We observe that some of the players are considered the dominant players on their teams, including Chris Paul and Kentavious Caldwell-Pope, who played an average of 34.5 minutes per game.

Using this measurement, a team can identify the players with a major impact on the team's offense, thus allowing the team's coaching staff to make better decisions when assembling their lineups. For example, in Figure 2, Kentavious Caldwell-Pope has the highest *Q-Ball* impact; in fact, he won the NBA championship with the LA Lakers during the 2019-2020 season as a part of the team's starting lineup. His statistics in the 2015-2016 season, however, were bad-to-mediocre compared to the league's average statistics. For example, his offensive rating was 107, which was close to the league average, and his field goals added (FGA), i.e., the number of extra points added based on the field goals made by the player compared to the average player, was -44.8, which was one of the worst in the league. During the 2019-2020 season, however, he was among the top 100 FGA players and the top 150 offensive rating players. In his scenario, we can see that the *Q-Ball* impact measure detected a strong indication of the this player's offensive potential.

Team Evaluation

The goal of this analysis is to determine *Q-Ball*'s ability to predict the teams' likelihood of conducting a successful offense (i.e., score points). In this experiment, we aim to evaluate each team's chances of a successful offense by calculating the average *Q-Ball* of each state and action for the entire team. We compare this measurement with the average number of points the team scored in all of the examined games and present the results in Figure 3.

Using the Pearson correlation, we measure the correlation between the team's average *Q-Ball* and the average number of points per game, and receive a coefficient of 0.622. We also measure the correlation between the team's offensive efficiency measurement (Oliver 2004), which is the average

Lineup	Avg *Q-Ball*	Possession Count	Avg Points Per Shot	Team
G.Monroe, J.Parker, K.Middleton, M.Carter-Williams, O.Mayo	1.690000	55	0.807527	Milwaukee Bucks
A.Iguodala, D.Green, F.Ezeli, S.Curry, S.Livingston	1.513313	70	0.886600	Golden State Warriors
A.Bradley, A.Johnson, I.Thomas, J.Crowder, K.Olynyk	1.265447	52	0.836962	Boston Celtics
A.Johnson, E.Turner, I.Thomas, J.Crowder, K.Olynyk	1.237501	63	0.890698	Boston Celtics
A.Iguodala, D.Green, H.Barnes, K.Thompson, S.Curry	1.177276	77	1.157662	Golden State Warriors

Table 1: The top five lineups with the highest average *Q-Ball*.

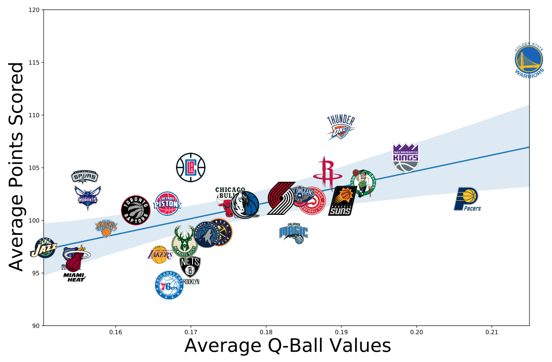

Figure 3: Points scored and average team *Q-Ball* comparison.

points scored by a team during 100 possessions and the average points scored, which produces a coefficient of 0.85. These results indicate the *Q-Ball* is able to accurately quantify the performance of a team by analyzing all of its actions (and not only those directly related to shooting). We conclude that *Q-Ball* is therefore capable of accurately modeling the quality of a team's offensive performance.

Q-Ball for Frame-by-Frame Analysis and Decision Support

The goal of this analysis is to determine *Q-Ball*'s ability to evaluate players' micro-actions and decision-making, as well as propose alternatives. Using our model, we are able to provide the *Q-Ball* value in a real-time, thus providing immediate feedback. We present an example for such a scenario in Figure 4. In this example, the Golden State Warriors (yellow) are playing against the Charlotte Hornets. The ball color changes according to its state, with blue representing a pass and red representing a shot.

The example above consists of multiple sequential decisions, such as passing and shooting the ball. Most of the time the *Q-Ball* value of shooting the ball is higher than the *Q-Ball* of other actions. This could be explained by the fact that NBA players are overall good shooters and that our reward equation only grants a positive reward for the shooting actions. In the example presented in Figure 4, we observe a relative difference in the *Q-Ball* for different actions. For example, we observe that in frames 2 and 4 the ball was passed to open players; the *Q-Ball* values of those passes are relatively higher than the action presented in frame 5, which is when a player drives to the hoop with three defenders close by. This indicates that the player's decision to drive to the hoop in frame 5 was incorrect.

Our model also allows us to compute the *Q-Ball* value for every possible action to determine whether a player made the right decision. Furthermore, by using the *Q-Ball* measurement in real time, we could evaluate the players' action in an actual game with high resolution. Thus, the model can inform the coaching staff when the players made good or poor decisions, and also assist in tactical decision-making. Moreover, our model enables simulations of alternative scenarios for every episode, thus enabling the teams to improving their strategies.

Simulating Game Scenarios. Our model can be used to synthesize data of alternative scenarios and grade it with *Q-Ball*. By evaluating the alternative proposed scenarios, we can determine whether a team or player made the best decisions during a given play, and recommend alternatives. In Figure 4, we present an example of the *Q-Ball* values of alternative scenarios suggested by our framework for the examined episode. In the figure, frame 7 present the original scenario as recorded during the game. Frame 7b presents the alternative scenario using simulated data.

In the example presented in Figure 4, at frame 6, #4 (Brandon Rush) decided to pass the ball to #23 (Draymond Green) who shot the ball; this action resulted in *Q-Ball* = 4.264. Instead of passing the ball, Brandon Rush could have shot the ball, as he was relatively close to the hoop; we simulated this action in frame 7b, the *Q-Ball* value of this action is *Q-Ball* = 4.321. When observing frame 7, we see that although Brandon Rush was surrounded by three defenders, none of them blocked his way to the hoop; frame 7b shows that those defenders obstructed the way to the hoop from Draymond Green, which led to a tough shot. The alternative scenario results in a *Q-Ball* that is in the 75th percentile of all shot values in the dataset, whereas the original shot *Q-Ball* was in the 50th percentile. This analysis emphasizes that the simulated scenario has a much better chance of resulting in scored points rather than missing the shot. Therefore, according to our model, Brandon Rush should have taken a shot instead of passing the ball.

The example described above demonstrates *Q-Ball*'s ability to produce specific recommendations for the complex scenarios of professional basketball. Using our approach, a team would not only be able to construct effective (or cost-effective) lineups, but also train the players how to perform when playing with (or against) specific players.

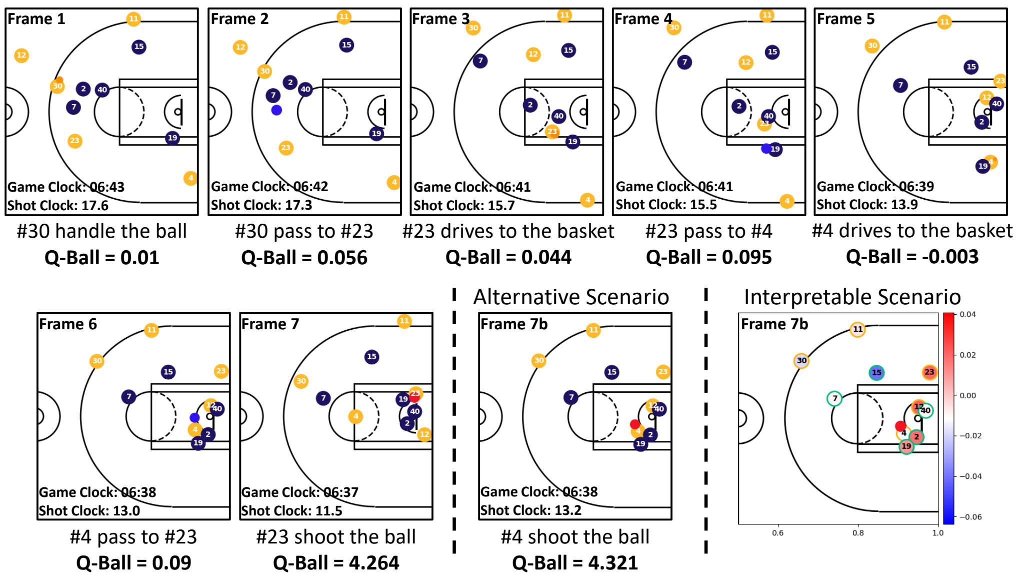

Figure 4: The *Q-Ball* value in a real-time evaluation (frames 1-7), alternative scenario (frame 7b), and demonstration of our ability to interpret the alternative scenario.

Interpreting the *Q-Ball* Values

The goal of this analysis is to interpret the *Q-Ball* values. This could help coaching staffs to improve their player actions by understanding the reason behind each suggestion our model provided. To interpret the *Q-Ball* values, we use the framework of SHAP, which we applied to our model. Using SHAP, we are able to calculate *SHAP values*, a metric that measures the contribution of each feature regarding the *Q-Ball* values. In Figure 4 we present the SHAP values of the alternative scenario (presented in Frame 7b). The positive values, which appear in red, indicate the locations that contributed to a higher positive *Q-Ball* value, and negative values, marked in blue, indicate the opposite.

From the results we observe that the locations of the attack players (marked with yellow) #23 and #12 have positive SHAP values, i.e., a positive impact on *Q-Ball*. This indicates that their positions increase the probability that the possession will result in a score. Using this evaluation we can also assess the positions of the defenders (marked with turquoise). We notice that player #15 has a negative SHAP value, i.e., its position has a negative contribution on *Q-Ball* value, which decreases the probability the attack will result in a positive result. This *Q-Ball* value makes sense because this player covers the pass range for two attack players, #11 and #30. In contrast to the position of the defenders #2 and #19, which should have been changed since their positions left player #4 uncovered in the paint area and player #23 completely unguarded. Using this analysis, we are able recommend better players placement.

We also explore the results of SHAP values for all features without the players' positions and noticed that the most contribute feature, with SHAP value of 0.125, is possession of the ball by player #4. Interestingly, this could indicate why player #4 should have continued to hold the ball instead of passing it to player #23 as observed in the original scenario in frame 7. We can also observe that the most negative feature is the game clock that has a negative SHAP value of -0.05, a value which indicates that this feature decreases the probability to score points at that specific game time. Using this evaluation we could present to the coaching staff which information contributed to higher *Q-Ball* values, thus, improving the players' decision making.

Limitations and Conclusions

Limitations. The main limitation of our approach is its dependence on data, which is a common drawback of data-driven methods. We will likely have to re-train the model to adjust it to new players and/or basketball leagues (e.g., the EuroLeague), which have not been observed before in our dataset (i.e., the cold-start problem). Moreover, we have currently not implemented a prospective evaluation, which is likely to require the collection of more data.

Conclusions. We present *Q-Ball*, a DRL algorithm for the analysis of basketball games. We use an extension of the DDPG architecture and show that it can be applied in an environment with both discrete and continuous actions. We use our proposed model to create a novel measurement, the *Q-Ball*, and show that it is well-suited for modeling the teams' shooting performance, players' actions, and their decision-making process. We demonstrate the average *Q-Ball* value correlates with the teams' scoring capabilities. Based on the *Q-Ball*, we can determine the optimal lineups for each team for each moment of the game. Our proposed approach has multiple practical applications, both in terms of strategic and tactical decision-making. As such, *Q-Ball* is an important first step in the integration of learning-based algorithms in all aspects of the game.

References

Bengio, Y.; Simard, P.; and Frasconi, P. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2): 157–166.

Decroos, T.; Bransen, L.; Van Haaren, J.; and Davis, J. 2019. Actions speak louder than goals: Valuing player actions in soccer. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1851–1861.

Hausknecht, M.; and Stone, P. 2015. Deep reinforcement learning in parameterized action space. *arXiv preprint arXiv:1511.04143*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Konda, V. R.; and Tsitsiklis, J. N. 2000. Actor-critic algorithms. In *Advances in neural information processing systems*, 1008–1014. Citeseer.

Kostrikov, I.; Agrawal, K. K.; Dwibedi, D.; Levine, S.; and Tompson, J. 2018. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *arXiv preprint arXiv:1809.02925*.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Liu, G.; Luo, Y.; Schulte, O.; and Kharrat, T. 2020. Deep soccer analytics: learning an action-value function for evaluating soccer players. *Data Mining and Knowledge Discovery*, 34(5): 1531–1559.

Liu, G.; and Schulte, O. 2018. Deep reinforcement learning in ice hockey for context-aware player evaluation. *arXiv preprint arXiv:1805.11088*.

Liu, G.; Zhu, W.; and Schulte, O. 2018. Interpreting Deep Sports Analytics: Valuing Actions and Players in the NHL. In *International Workshop on Machine Learning and Data Mining for Sports Analytics*, 69–81. Springer.

Lundberg, S.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*.

Macdonald, B. 2020. Recreating the Game: Using Player Tracking Data to Analyze Dynamics in Basketball and Football. *Issue 2.4*, 2(4).

Manner, H. 2016. Modeling and forecasting the outcomes of NBA basketball games. *Journal of Quantitative Analysis in Sports*, 12(1): 31–41.

Neiman, T.; and Loewenstein, Y. 2011. Reinforcement learning in professional basketball players. *Nature communications*, 2(1): 1–8.

Oliver, D. 2004. *Basketball on paper: rules and tools for performance analysis*. Potomac Books, Inc.

Sampaio, J.; McGarry, T.; Calleja-González, J.; Jiménez Sáiz, S.; Schelling i del Alcázar, X.; and Balciunas, M. 2015. Exploring game performance in the National Basketball Association using player tracking data. *PloS one*, 10(7): e0132894.

Seidl, T.; Cherukumudi, A.; Hartnett, A.; Carr, P.; and Lucey, P. 2018. Bhostgusters: Realtime interactive play sketching with synthesized nba defenses. In *12 th Annual MIT Sloan Sports Analytics Conference*.

Sicilia, A.; Pelechrinis, K.; and Goldsberry, K. 2019. Deephoops: Evaluating micro-actions in basketball using deep feature representations of spatio-temporal data. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2096–2104.

Torres, R. A. 2013. Prediction of NBA games based on Machine Learning Methods. *University of Wisconsin, Madison*.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Vračar, P.; Štrumbelj, E.; and Kononenko, I. 2016. Modeling basketball play-by-play data. *Expert Systems with Applications*, 44: 58–66.

Wang, J.; Fox, I.; Skaza, J.; Linck, N.; Singh, S.; and Wiens, J. 2018. The advantage of doubling: a deep reinforcement learning approach to studying the double team in the NBA. *arXiv preprint arXiv:1803.02940*.

Wang, K.; Li, H.; Gong, L.; Tao, J.; Wu, R.; Fan, C.; Chen, L.; and Cui, P. 2020. Match Tracing: A Unified Framework for Real-time Win Prediction and Quantifiable Performance Evaluation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2781–2788.

Watkins, C. J.; and Dayan, P. 1992. Q-learning. *Machine learning*, 8(3-4): 279–292.

Zhang, S.; and Sutton, R. S. 2017. A deeper look at experience replay. *arXiv preprint arXiv:1712.01275*.