DeepHoops: Evaluating Micro-Actions in Basketball Using Deep Feature Representations of Spatio-Temporal Data

Anthony Sicilia
School of Computing & Information
University of Pittsburgh
anthonysicilia@pitt.edu

Konstantinos Pelechrinis
School of Computing & Information
University of Pittsburgh
kpele@pitt.edu

Kirk Goldsberry
School of Business
University of Texas, Austin
kpg447@utexas.edu

ABSTRACT

Basketball is one of a number of sports which, within the past decade, have seen an explosion in quantitative metrics and methods for evaluating players and teams. However, it is still challenging to evaluate individual off-ball events (e.g., screens, cuts away from the ball etc.) in terms of how they contribute to the success of a possession. In this study, we develop a deep learning framework (*DeepHoops*) to process a unique dataset composed of spatio-temporal tracking data from NBA games in order to generate a running stream of predictions on the expected points to be scored as a possession progresses. We frame the problem as a multi-class sequence classification problem in which our model estimates probabilities of *terminal actions* taken by players (e.g. take field goal, turnover, foul etc.) at each moment of a possession based on a sequence of ball and player court locations preceding the said moment. Each of these terminal actions is associated with an expected point value, which is used to estimate the expected points to be scored. One of the challenges associated with this problem is the high imbalance in the action classes. To solve this problem, we parameterize a downsampling scheme for the training phase. We demonstrate that *DeepHoops* is well-calibrated, estimating accurately the probabilities of each terminal action and we further showcase the model's capability to evaluate individual actions (potentially off-ball) within a possession that are not captured by boxscore statistics.

1 INTRODUCTION

While data have been an integral part of sports since the first boxscore was recorded in a baseball game during the 1870s, it is only recently that machine learning has really penetrated the sports industry and has been utilized for facilitating the operations and decision making of sports franchises. One main reason for this

is our current ability to collect more fine-grained data; data that capture (almost) everything that happens on the court/field. For instance, since the 2013-14 season, the National Basketball Association (NBA) has mandated its 30 teams to install an optical tracking system that collects information 25 times every second for the location of the ball and all the players on the court. These data are further annotated with other information such as the current score, the game and shot clock time etc. Optical tracking data provide a lens to the game that is much different from traditional player and team statistics. Many actions that can affect the outcome of a game/possession happen away from the ball (off-ball actions) and are not recorded in boxscore-like metrics that capture almost exclusively on-ball events. For instance, while boxscore statistics such as steals and blocks do not accurately reflect the defensive performance of a team, analyzing the spatial movement of players can provide better insights in the (individual and team) defensive ability [10]. As another example, the Toronto Raptors - who were among the first teams to make use of this technology - were able to identify optimal positions for the defenders, given the offensive formation [20].

Micro-actions refer to individual basketball moves by players, that combined, create the team's offensive and/or defensive execution. A screen, a pass, or a backdoor cut are examples of such micro-actions. These micro-actions, although important for the successful execution of a team's game plan, are rarely evaluated. Even when we have volume statistics from these actions the context is usually absent, which makes it hard to evaluate the importance of each of these actions. For example, simply counting passes can be very misleading; a pass exchange in the backcourt is certainly not as valuable as a pass that leads to an open three point shot. In order to achieve our goal we use a unique dataset from 784 NBA games from the 2016-17 regular season that includes the aforementioned information captured by the optical tracking system. We build a deep learning framework, namely *DeepHoops*, that predicts the running probability distribution of **terminal actions** conditional to the players' and the ball's spatio-temporal sequence over a time window W of length T. Here, terminal actions refer to actions that lead to change or reset of ball possession. We consequently transform this distribution to an expected points value ϵ_π that essentially captures the expected points to be scored from the offense during this possession (within r seconds from now). *DeepHoops* also obtains an embedding for the players that is fed into the classifier hence, accounting for the personnel on court. This is particularly important since the same offensive scheme could have very different outcomes depending on who is on the court.

With *DeepHoops* providing an expected points value $\epsilon_\pi(\tau)$ at any time τ during the possession, we can calculate the expected
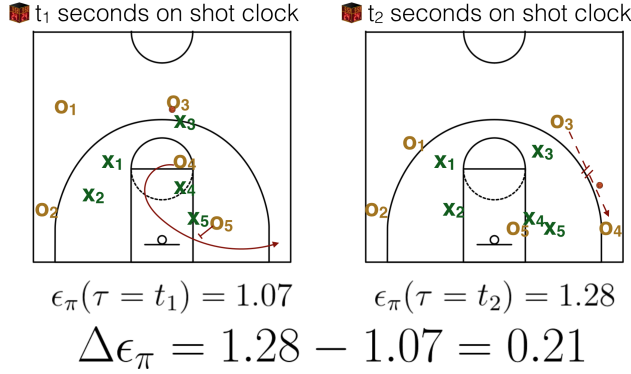
$$\epsilon_\pi(\tau = t_1) = 1.07 \qquad \epsilon_\pi(\tau = t_2) = 1.28$$

$$\Delta\epsilon_\pi = 1.28 - 1.07 = 0.21$$

Figure 1: *DeepHoops* is able to capture the expected points to be scored by the offense for a given possession snapshot. This allows us to estimate how micro-action(s) affect the scoring chances of a team. In the example above, the off-ball movement of player O_4 in conjunction with the screen by O_5 and the pass from O_3 increased the expected points from 1.07 to 1.28.

points added by a micro-action α at time τ as $\Delta\epsilon_\pi(\alpha, \tau) = \epsilon_\pi(\tau + \varepsilon) - \epsilon_\pi(\tau - \varepsilon)$. As an illustrative example, Figure 1 presents two snapshots from an offensive set. A player of the offense (O_4) moves towards the left corner three area and receives a screen from O_5. He eventually gets the pass from the ball handler, O_3, and takes an open shot. During the first snapshot at time t_1, the expected points to be scored are $\epsilon_\pi(\tau = t_1) = 1.08$. After the screen and pass for an open corner three, this increases to $\epsilon_\pi(\tau = t_2) = 1.27$, giving a $\Delta\epsilon_\pi = 0.21$. With this approach, we can start dissecting a possession to its smaller actions and quantify the overall impact of these actions to the expected final outcome. Furthermore, the applications of ***DeepHoops*** extend to "what-if" scenarios as well. What should we have expected to happen if O_3 did not pass to O_4 for the open shot, but drove to the basket since there is an open lane in the middle with X_4 and X_5 trying to close to the corner ? Will a different decision by the players on the court have led to higher expected points for the possession? This can drive evaluations of on court decisions by the players. Similar evaluations can happen for the defense. What would be the value of $\epsilon_\pi(\tau = t_2)$ if X_5 had hedged the screen better ?

Contributions: Our study contributes to the increasing literature on sports analytics by providing a generic deep learning framework that can quantify the value of micro-actions in basketball. As we will discuss in the following section, our work utilizes deep neural networks to essentially build on, and expand, the Markov chain model developed by Cervone *et al.* [4] to estimate the expected points value. ***DeepHoops*** does not require any type of feature selection - e.g., definition and modeling of transitions between states - but rather obtains as input the raw trajectories of all the players and the ball. It consequently learns the most predictive features for the terminal action of a possession. Furthermore, it accounts for the players who are on the court through a multi-dimensional embedding. ***DeepHoops*** could also serve as a blueprint for similar models of other *fluid* sports (e.g., soccer, American football, volleyball etc.).

Notation/Term	Description
ϵ_π	Expected Points
$\Delta\epsilon_\pi$	Expected Points Added
$\boldsymbol{x}_t^{(i)}$	A moment during possession i capturing the locations of players and the ball at time t
$\boldsymbol{s}^{(i)}$	Players on court during possession i
$W_\tau^{(i)}$	A temporal sequence window of moments during possession i that ends at time τ
T	Length of temporal window W
r	Blind spotÏĎ (buffer) at the end of window W
K	Downsampling parameter
α	Player micro-actions within play (pass, cut, screen etc.)
e	Terminal action which ends a possesion (e.g., shot attempt, turnover etc.)
$\mathcal{L}$	Set of terminal actions

Table 1: Notations used throughout the paper.

The rest of the paper is organized as follows: Section 2 discusses work relevant to our study and provides the required technical background, while Section 3 presents the architecture of ***DeepHoops***. Section 4 presents the evaluation of our method, while Section 5 concludes and discusses the scope and limitations of our study.

2 BACKGROUND AND RELATED WORK

In this section we will provide some background on the neural network components we will include in the ***DeepHoops*** architecture. We will also discuss relevant studies on basketball analytics and more specifically on the use of spatio-temporal data in sports.

2.1 Neural Networks

There are primarily two neural network components included in the ***DeepHoops*** architecture: a Long-Short Term Memory Network (LSTM) and a Neural Embedding.

Long-Short Term Memory Network: The LSTM network, originally introduced by Hochreiter and Schmidhuber [17] and later modified by Gers *et al.* [12] is a type of recurrent neural network (RNN) meant to solve the vanishing gradient problem [14]. In general, these networks are capable of processing sequential data by using the current input as well as the previous state to compute the current state [14]. For example, in the sports analytics domain, they have been used to process spatio-temporal data similar to ours for the task of offensive play-call classification [32]. RNNs like other neural network models are thought to be able to represent complex functions more capably when designed with a deeper (multilayer) structure and there is empirical evidence to support this as well [14, 15]. This multilayer structure can be accomplished by *stacking* the output of each subsequent layer to form a stacked recurrent neural network (sRNN) [27]. We employ a stacked LSTM for the task of sequence modelling with the specifics discussed in detail in Section 3.

Neural Embedding: When representing discrete objects as input to a neural network, it is often beneficial to embed the object in some latent space such that similar objects are *close* to each other in this space. For example, word embeddings have been widely used in the field of natural language processing for tasks such as

sentence classification [18]. In the basketball setting, Wang and Zemel [32] used an autoencoder with the shooting tendencies of a player as input in order to identify a latent representation for the players. This was then used to identify the position/role of a player based on his neighbors in the space. An embedding may also be learned during the training phase (with the other network parameters) as can be the case with word embedding [18]. This is the approach we take in **DeepHoops** where we employ a player embedding learned in tandem with the end-to-end architecture. In this way, two players will be close in the latent space based on their *impact* on the probability distribution of the terminal actions. We discuss this player embedding in detail in Section 3. Finally, Table 1 presents some of the mathematical notations we use in the rest of the paper.

2.2 Related Literature

Player tracking data and basketball analytics: The availability of optical tracking sports data has allowed researchers and practitioners in sports analytics to analyze and model aspects of the game that were not possible with traditional data. For instance, as alluded to above, Franks *et al.* [9] developed models for capturing the defensive ability of players based on the spatial information obtained from optical tracking data. Their approach is based on a combination of spatial point processes, matrix factorization and hierarchical regression models and can reveal information that cannot be inferred with boxscore data. As an example, the proposed model can identify whether a defender is effective because he reduces the quality of a shot or because he reduces the frequency of the shots altogether. Furthermore, Miller *et al.* [23] use Non-Negative Matrix Factorization to reduce the dimensionality of the spatial profiles of player's shot charts. The authors use a log-Gaussian Cox point process to smooth the raw shooting charts, which they show provides more intuitive and interpretable patterns. Papalexakis *et al.* [26] extended this approach using tensor factorization methods to build shooting profiles that can account for contextual information such as, shot clock, score differential etc. Additionally, using Bezier curves and Latent Dirichlet Allocation, a dictionary for trajectories that appear in basketball possessions was developed in [24]. In a different direction, Cervone *et al.* [3] computed the court's Voronoi diagram based on the players' locations and formalized an optimization problem that provides court realty values for different areas of the court. This further allowed the authors to develop new metrics for quantifying the spacing and the positioning of a lineup/team.

Using optical tracking data, Yue *et al.* [33] further developed a model with conditional random fields and non-negative matrix factorization for predicting the near-term actions of an offense (e.g., pass, shoot, dribble etc.) given its current state. In a tangential direction, D'Amour *et al.* [8] develop a continuous time Markov-chain to describe the discrete states a basketball possession goes through. Using this model the authors then propose entropy-based metrics over this Markov-chain to quantify the ball movement through the unpredictability of the offense, which also correlates well with the generation of opportunities for open shots. Seidl *et al.* [30] further used optical tracking data to learn how a defense is likely to react to a specific offensive set using reinforcement learning. Recently, a volume of research has appeared that utilizes

deep learning methods to analyze spatio-temporal basketball data, learn latent representations for players and/or teams, and identify and predict activities [22, 34], while additionally, Daly-Grafstein and Bornn [7] used the actual trajectory of a ball during a shot to obtain a robust estimation of the shooting skill of a player using a smaller sample of shots.

Close to our study is the work from Harmon *et al.* [16], who utilized player trajectory and a convolutional neural network to predict whether a shot will be made or not. However, they only focus on possessions that end with a shot and they do not consider the development of the possession. The closest work to our study is that of Cervone *et al.* [4] who utilized optical tracking data and developed a model for the expected possession value (EPV) using a multi-resolution stochastic process model. Our study builds on this work, providing a general framework which does not require the definition of features or possession states and can also be adjusted for applications in other sports. There are a few subtle but significant differences between **DeepHoops** and the work from Cervone *et al.* [4]. In particular, in our work the expected points are calculated relative to a temporal window instead of the possession as a whole. Hence, while [4] is more of a prediction model of what the offense will do next and how this will affect the end result of the possession, **DeepHoops** is more of a tool for value attribution given sequences that led to *good/bad* outcomes. **DeepHoops** also includes a finer granularity for a possession's terminal actions (e.g., including classes for fouls, shooting fouls etc.).

Other sports: While basketball is the sport that has been studied the most through optical tracking data there is relevant literature studying other sports as well. For example, Bialkowski *et al.* [1] formulate an entropy minimization problem for identifying players' roles in soccer. They propose an EM-based solution, which is able to identify the players' roles and the formation of the team using player tracking data. Lucey *et al.* [21] also used optical tracking data to predict the probability of scoring a goal by extracting features that go beyond just the location and angle of the shot. More recently, Le *et al.* [19] develop a collaboration model for multi-agents using a combination of unsupervised and imitation learning. They further apply their model to optical tracking data from soccer to identify the optimal positioning for defenders given a particular formation of the offense, thus, allowing teams to evaluate the defensive skills of individual players. In a tangential effort, Power *et al.* [28] define and use a supervised learning approach for the risk and reward of a specific pass in soccer using detailed spatio-temporal data. The risk/reward of a specific pass can further quantify offensive and defensive skills of players/teams. While we introduce **DeepHoops** as a framework for analyzing basketball data, it should be evident that it can really be used to analyze spatio-temporal (and in general multi-aspect) data for other sports as well.

3 PROPOSED METHOD

In this section we formally define our problem and outline the details of our proposed solution. We begin with a discussion of the spatio-temporal tracking data, followed with the formal definition of our notion of expected points, which will be the medium through which we assign value to micro-actions. We then provide the details behind the **DeepHoops** architecture utilized to learn a deep feature

representation of the spatio-temporal data, and consequently, estimate the expected points. Finally, we describe a technique used in the training phase to remedy the problem of highly unbalanced training data.

3.1 Description and Processing of Spatio-Temporal Data

As aforementioned, to build **DeepHoops** we use a unique dataset composed of 784 NBA games from the 2016-17 regular season. Of primary interest is the optical tracking data which represents the NBA court as a three dimensional coordinate system in which the court location of offensive and defensive players can be expressed using rectangular coordinates along the (x, y)-plane and the court location and height of the ball can be expressed by taking into account the third dimension as well. During each second of gameplay, this information is recorded 25 times at evenly spaced intervals.

The data is extensively annotated allowing for labeling of specific events and possession outcomes. Using this annotation it is simple to segment the game into well-defined team offensive possessions (i.e. into segments during which one team maintains control of the ball with the intention of scoring). Additionally, the data provide information about the specific players that are on the court as well as how much time remains on the shot-clock (which regulates the remaining maximum length of the possession).

Possessions: Our dataset consists of a sample of more than 134,000 team possessions of interest. We define a possession i as a sequence of n **moments** $(x_t^{(i)})_{t=1}^n$ where each moment is a 24-dimensional vector, i.e., $x_t^{(i)} \in \mathbb{R}^{24}$. The first 20 elements capture the court location of the 10 players via (x, y)-coordinates, the next three represent the court location and height of the ball via (x, y, z)-coordinates, while the last element represents the current value of the shot-clock. Each of these moments are well-annotated with event descriptions when relevant (e.g., a pass occurred during moment $(x_\tau^{(i)})$). Additionally, each possession i maintains information about the 5 offensive players $\{s_j^{(i)}\}_{j=1}^5$ and 5 defensive players $\{s_j^{(i)}\}_{j=6}^{10}$ on the court via one-hot encoding. Furthermore, we convert the court locations into polar coordinates with respect to the current offensive team's basket, since this way we are also encoding the *target* location and are likely more informative.

Temporal Window: To estimate the expected points for a possession, we define a temporal window to act as the spatio-temporal input to **DeepHoops**. Specifically, for possession i and moment τ, we define a window of length T as the subsequence:

$$W_\tau^{(i)} = (x_t^{(i)}) \quad \text{with} \quad \{\tau - (T + r) \le t \le \tau - r\} \quad (1)$$

A window defined at moment τ captures T moments of a possession leading up to the time of interest (i.e., τ) with a buffer or blind spot r. As we will describe in the following, each of these windows will be assigned an outcome label $y_\tau^{(i)}$, which corresponds to a possession terminal action. The blind spot intuitively serves the purpose of pruning information late in the window, which are trivially indicative of a terminal action. Furthermore, restricting to smaller (sliding) windows, instead of using the entirety of the trajectories from the beginning of the possession makes more sense in the basketball setting [16]. Most plays develop over a small period of time

(often less than 5 seconds) and if they are not successfully executed (e.g., finding an open shot) the offense resets its scheme. Hence, multiple plays may develop and fail during the same possession, so the temporal window effectively restricts the sample to only the information of primary interest preceding a given moment. For this reason, unless otherwise noted we will use T = 128 moments (just over 5 seconds) with $r = 16$ moments (just over half of a second).

Outcome Labels: As aforementioned, each window is labeled with an outcome $y_\tau^{(i)}$ that represents the type of (possession) terminal action that occurred at the end of the window. If the terminal action of the possession (e.g. a field goal attempt, a turnover etc.) occurs at moment τ, then $W_\tau^{(i)}$ is labeled with this action, otherwise it is labeled as **null**. The set of labels $\mathcal{L}$ consists of:

- **Field Goal Attempt**: The window ends with a field goal (FG) attempt, made or missed.
- **Shooting Foul**: The window ends with a foul (illegal defensive action) during a shot attempt that awards free throw attempts to the player who was fouled.
- **Non-Shooting Foul**: The window ends with a player committing a non-shooting foul.
- **Turnover**: The window ends with the offense committing a turnover (e.g., out-of-bounds pass, offensive foul etc.), giving up possession of the ball to the opponent.
- **null**: No terminal action was recorded at the end of the window, and hence, the possession is still in progress.

The inclusion of a **null** label is crucial. As alluded to above an offensive set is usually executed over a short time period and then resets if needed. Therefore, labeling a whole possession, instead of sliding windows, would result in extensive noise in the training set which masks vital information. For example, let us consider a possession where the offense initially set up a screen that was ineffective. The offense had to reset, and after executing a different scheme, they took (and made) a corner three-point shot. If we only had a single label for this possession, then the ineffective screen action would be labeled with a terminal action that adds value (i.e., a FG attempt). However, with our labeling scheme - i.e., the use of the sliding temporal window - the ineffective screen will be included within a window labeled as **null**, as it should, since it did not lead to an immediate shot/score or turnover. With this said, the label space $\mathcal{L}$ exhibits a very high imbalance, since the majority of the windows end with no event. The **null** class out-weights all other class labels roughly 600 to 1. This class imbalance requires careful attention in the training phase. To handle this, we parameterize a downsample of the majority class (details discussed in Section 3.3).

3.2 Translating Labels to Points Per Possession

Given a possession i we can begin to define (a) the expected points $\epsilon_\pi^{(i)}(\tau)$ computed at each moment τ within the possession, as well as, (b) its application for valuing micro-actions within possessions.

Expected Points: Every possession has a baseline expected points β_π to be scored. We can calculate this by dividing the total number of points scored over all games with the number of total possessions. In our dataset, $\beta_\pi = 1.02$. Using **DeepHoops** we can further adjust this value - in real time - based on the probability of the labels in $\mathcal{L}$. In particular, we define a function $v : \mathcal{L} \to \mathbb{R}$ that maps every class/terminal action to a number that captures the

points above expectation (i.e., above β_π) to be scored if this terminal action is realized during the play. For example, the average points scored during a possession that terminates with a shot (average points per shot) is 1.25. Hence, a possession that ends in a FG attempt is worth $1.25 - \beta_\pi = 1.25 - 1.02 = 0.23$ points *above* expectation, and therefore, we define ν(FG Attempt) = 0.23. Figure 2 presents the mapping for every terminal action to points above β_π. For example, a turnover that ends in a change of possession has a value of $\nu = -\beta_\pi$ (because this terminal action counteracts any chance of scoring), while the **null** class has a value of 0 (because this terminal action provides no new information with which to adjust the baseline). For the non-shooting fouls, we have incorporated the fraction of fouls committed while the defense is in the penalty (because in this situation free throws are awarded). Note here that if we are interested in a specific possession, we will know whether the defense is in the penalty and we can incorporate this information directly.
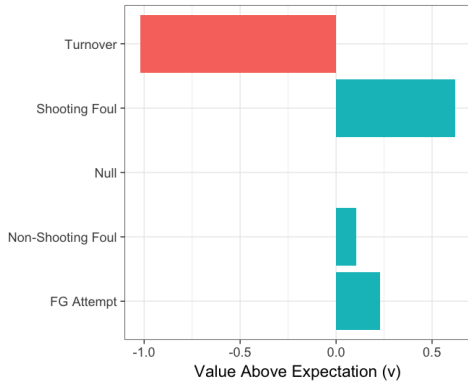
Figure 2: For each outcome, we calculate the points above $\beta_\pi = 1.02$ the offense will gain if this outcome is realized.

The expected points $\epsilon_\pi{}^{(i)}(\tau)$ for possession i at time τ can now be defined as an expectation above β_π conditional to (i) the temporal window $W_\tau^{(i)}$, and (ii) the players on the court. Specifically, if y is the discrete random variable that captures the outcome of window $W_\tau^{(i)}$ and follows the distribution $P(\text{y})$, then $\epsilon_\pi{}^{(i)}(\tau)$ is defined as:

$$\epsilon_\pi{}^{(i)}(\tau) = \beta_\pi + \mathbb{E}_{\text{y} \sim P(\text{y})}[\nu(y)|W_\tau^{(i)}, \{s_j^{(i)}\}_{j=1}^{10}] \qquad (2)$$

For example, in the extreme (and rather unrealistic) case where all labels have a zero probability except the turnover that has a 100% probability, the expected points will be equal to: $\epsilon_\pi{}^{(i)}(\tau) = 1.02 - 1 \cdot 1.02 = 0$, as it should be since the team will not score and the ball will change possession.

Expected Points Added: The primary goal of defining ϵ_π is to assign a value to specific micro-action α as demonstrated in Figure 1. Intuitively, the inclusion of a valuable micro-action within a temporal window $W_\tau^{(i)}$ during possession i should alter the distribution of outcomes, making higher value outcomes more likely conditional to the new window. Formally, if α occurs at moment τ within a possession i, the expected point added can be calculated as:

$$\Delta\epsilon_\pi(\alpha, \tau)^{(i)} = \epsilon_\pi{}^{(i)}(\tau + \varepsilon) - \epsilon_\pi{}^{(i)}(\tau - \varepsilon) \qquad (3)$$

where $\varepsilon > r$ to ensure that the micro-action α occurs in the new window $W_{\tau+\varepsilon}^{(i)}$. Notice, that this definition, in conjunction with Equation 1, imply that taking $\varepsilon = \delta + r$ corresponds to a shift around the micro-action of δ moments.

3.3 The *DeepHoops* Architecture

Based on the discussion above, the problem of calculating expected points and expected points added translates to estimating the probability distribution: $P(y|W_\tau^{(i)}, \{s_j^{(i)}\}_{j=1}^{10})$. We model the probability of each label with a softmax function, that is, we estimate the probability for label y_i as:

$$P(y|W_\tau^{(i)}, \{s_j^{(i)}\}_{j=1}^{10}) = \frac{e^{z_i}}{\sum_{y_j \in \mathcal{L}} e^{z_j}} \qquad (4)$$

where z_i is computed using the learned feature representation provided by the neural network g with parameters Θ:

$$z = W^{\text{T}} g(W_\tau^{(i)}, \{s_j^{(i)}\}_{j=1}^{10}; \Theta) + b \qquad (5)$$

Here, W and b correspond to the linear layer producing the log outcome probabilities, and Θ are the parameters of g optimized to learn a joint feature representation of $W_\tau^{(i)}$ and $\{s_j^{(i)}\}_{j=1}^{10}$ such that the negative log-likelihood is minimized [14]. Formally, in expectation over the data distribution, that is:

$$\min_{W, b, \Theta} -\mathbb{E} \sum_{y_j \in \mathcal{L}} \mathbf{1}_{\{y_j\}} \log P(y_j|W_\tau^{(i)}, \{s_j^{(i)}\}_{j=1}^{10}) \qquad (6)$$

where $\mathbf{1}_{\{*\}}$ is an indicator function. The remainder of this section is devoted to describing the network g (Figure 3) that forms the core component of **DeepHoops**.

To jointly learn a feature representation of $W_\tau^{(i)}$ and $\{s_j^{(i)}\}_{j=1}^{10}$, we construct g with two main modules. The first is a sequence processing module and uses a stacked LSTM network to learn a representation of the spatio-temporal window leading up to time τ. This module allows for important information about on-court actions to be captured as the play progresses. The second module primarily serves the purpose of capturing information with regards to who is on the court. This additional information is meant to model the impact of the specific lineup playing on the expected value of the possession as discussed in Section 1. The output of the two main components is combined and fed into a dense (i.e. fully connected) layer to generate the final joint feature representation.

Stacked LSTM Network: The LSTM network is a common, effective neural network for processing sequential inputs [14, 15]. Each layer of the LSTM has a number of LSTM cells (in our implementation, we have 32 for each of the 3 layers). The state of the ith cell at timestep t is represented by the ith element of vector $h^{(t)}$. Here, the timesteps run over the length of our sequential data. Each of these cells processes data similarly to the standard unit of an RNN, that is, they compute an external state (i.e. $h^{(t)}$) via information from previous states and the current input. However, there is an additional internal state $c^{(t)}$ and a number of gates that control the flow of information [14] (see Figure 3). In our implementation [6], given a window, which we rewrite for clarity as $W_\tau = (x^{(1)}, \ldots x^{(\text{T})})$, the computation for cell i at time step t can be written as:
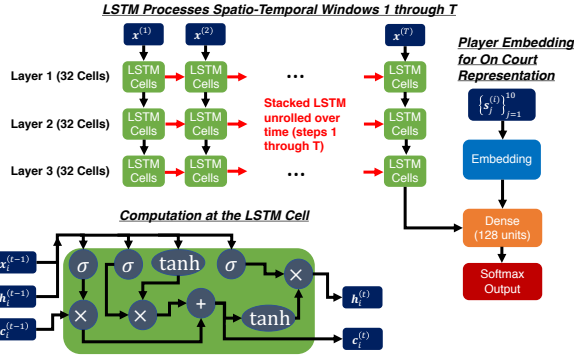
Figure 3: Diagram of *DeepHoops*. The LSTM network (displayed unrolled) learns a feature representation of the spatio-temporal window. A latent space representation of on court players is concatenated and processed by an additional dense layer before softmax output for probability estimation.

$$i_i^{(t)} = \sigma(b_i' + \sum_j U_{i,j}' x_j^{(t)} + \sum_j W_{i,j}' h_j^{(t-1)}) \tag{7}$$

$$f_i^{(t)} = \sigma(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}) \tag{8}$$

$$c_i^{(t)} = f_i^{(t)} c_i^{(t-1)} + i_i^{(t)} \tanh(b_i^c + \sum_j U_{i,j}^c x_j^{(t)} + \sum_j W_{i,j}^c h_j^{(t-1)}) \tag{9}$$

$$o_i^{(t)} = \sigma(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}) \tag{10}$$

$$h_i^{(t)} = o_i^{(t)} \tanh(c_i^{(t)}) \tag{11}$$

where in the above all subscripts are component indices and the U^*, W^*, b^* are the parameters to be learned. Conceptually, i is the input gate, f is the forget gate, and o is the output gate. Each of these gates controls what information is carried to the next time step (via internal state c) and what information is output (via external state h) [12, 14]. As mentioned *DeepHoops* uses a stacked LSTM [15, 27] which simply treats the external state of the previous layer as input to the next. Our stacked LSTM is restricted to processing fixed-length sequences (i.e. our windows of length T), so the final state of a given layer is referenced as $h^{(T)}$. We therefore define the output of our LSTM network as the final state $h^{(T)}$ of the last layer which will be concatenated with the player representations (see below) before processing by a final dense layer (Figure 3).

Player Representation: As aforementioned, we jointly learn an embedding of individual players within our end-to-end architecture. The players are represented through vectors that are updated with each training sample. This update is formally dependent on its contribution to the output of the network g. Namely, the embedding implementation [6] randomly initializes a group of dense d dimensional vectors $a^j \in \mathbb{R}^d$ to represent player $j \in \{1, \ldots, m\}$. If we form the tensor $A = [a^1, \ldots, a^m] \in \mathbb{R}^{d \times m}$, then the one-hot encoding of the jth player $s_j^{(i)}$ can be used to extract that player's representation via the matrix multiplication $As_j^{(i)}$. As discussed, the

extracted player representations are concatenated to the final state $h^{(T)}$ of the last layer of the LSTM and fed through an additional dense layer before classification. This way, the player representations are jointly learned in the sense that whenever a player is present on court, his randomly initialized vector is updated during backpropagation to minimize the loss (Equation 6) for the classification task (just as all other weights within the network g). Players with *similar* contributions to the distribution of terminal actions are then expected to be close in the corresponding latent space.

Downsampling Scheme: With the network g defined, we now address the imbalance of the **null** label. As mentioned in Section 3.1, these **null** temporal windows capture important information about micro-actions, but training on such highly imbalanced data could pose problems in generating well calibrated probabilities. One technique when dealing with high class imbalance is to down-sample the majority class [5]. We propose and use a problem-specific down-sampling technique which, importantly, ensures that each *possession* has equal representation in the newly generated training set. In particular, from a possession i, we uniformly sample K time-steps $\{t_{i,k}\}_{k=1}^K$ for which the corresponding window $W_{t_{i,k}}^{(i)}$ is labeled as **null**. Then, besides the single window $W_\tau^{(i)}$ which is labeled by this possession's terminal action, all other windows are discarded. Hence, during the training phase, we compute the loss L for a batch of size N possessions with a down-sample of size K as:

$$L_i = \sum_{y_j \in \mathcal{L}} \mathbf{1}_{\{y_j\}} \log P(y_j | W_\tau^{(i)}, \{s_j^{(i)}\}_{j=1}^{10}) \tag{12}$$

$$L_i' = \sum_{k=1}^K \sum_{y_j \in \mathcal{L}} \mathbf{1}_{\{y_j\}} \log P(y_j | W_{t_{i,k}}^{(i)}, \{s_j^{(i)}\}_{j=1}^{10}) \tag{13}$$

$$L = -\frac{1}{N(K+1)} \sum_{i=1}^N L_i + L_i' \tag{14}$$

This technique is naturally parameterized by K and we explore the impact of this parameter in Section 4. We additionally re-generate the down-sampled dataset on each epoch (i.e. the sample $\{t_{i,k}\}_{k=1}^K$ is redrawn for each possession on any new epoch).

4 EXPERIMENTS

For our evaluations, we break the data into training, validation, and test sets at a 75-10-15 split. All results presented are on the test set, while the the performance on the validation set is monitored in training to prevent overfitting via Early Stopping [29]. The monitored metric is the Brier score (discussed in what follows) with a minimum required improvement of 0.01 over 5 epochs. Only the *best* performing model on the validation set is used in the testing process. For all experiments, we used the following hyperparameters for the *DeepHoops* architecture: 3 LSTM layers with 32 cells each, an embedding of dimensionality $d = 8$, and 1 dense layer with 128 units prior to classification. To prevent overfitting [31], dropout with rate of 0.3 is applied to the output of the dense layer and variational dropout [6, 11] is applied throughout the LSTM network with rate of 0.2 for transformation of both input and recurrent state.

4.1 Probability Calibration

Our main objective is to evaluate the accuracy of the predicted probabilities for the possible possession outcomes as this is directly indicative of the accuracy of the expected points. Evaluation of the quality of a probability, i.e., probability calibration, has traditionally been done through the Brier score and the reliability curves.

Brier Score: The Brier score is a proper scoring rule that quantifies the calibration of a set of probabilistic predictions [2, 13, 25]. If we have N samples (i.e., predictions) and R possible outcomes, the Brier score is calculated as:

$$BS = \frac{1}{N} \sum_{i=1}^{N} \sum_{r=1}^{R} (f_{i,r} - o_{i,r})^2 \qquad (15)$$

where $f_{i,r}$ is the model predicted probability for outcome/label r on sample i and $o_{i,r}$ is 1 if sample i's label is r, while it is 0 otherwise. From the above definition it should be evident that between two models, the one with the lowest Brier score is better calibrated.

Typically the Brier score of a model is compared to a *climatology* model which assigns to every outcome its baseline probability, i.e., the historical frequency of occurrences for each outcome [2, 25]. Using the Brier score of this climatology model BS_{ref}, we can further calculate the *skill* of probability estimates using the Brier Skill Score (BSS) [13, 25]. Specifically, if BS_{ref} is the Brier score of the climatology model, then BSS is calculated as:

$$BSS = 1 - \frac{BS}{BS_{ref}} \qquad (16)$$

BSS will be equal to 1 for a model with perfect calibration, i.e., $BS = 0$. A model with no *skill* over the climatology model, will have a value of 0 since $BS = BS_{ref}$. If $BSS < 0$, then the model exhibits less skill than even the reference model.

Table 2 displays the Brier score of **DeepHoops**, the corresponding climatology model, as well as, the Brier Skill Score of **DeepHoops** for different values of the downsampling parameter K. We also include the minimum training time for an epoch of **DeepHoops** for different values of K. As can be seen, regardless of the value of K, **DeepHoops** exhibits better calibration as compared to that of the climatology model. Additionally, by comparing the Brier Skill Score, we see that improvement over the climatology model is best when $K = 2$. Furthermore, as one might have expected, higher value of K requires longer time to train. For this reason, in the remainder of the experiments, we pick $K = 2$.

Reliability Curves: In order to estimate the accuracy of the output probabilities for each label, ideally we would like the same window within a possession to *replay* several times and then estimate the number of times that it ended up to each possible outcome. Obviously this is not possible and hence, we will rely on the reliability curves. In particular, if the predicted probabilities were accurate, when considering all the windows where terminal action e was predicted with a probability of x%, then terminal action e should have been observed in (approximately) x% of these windows. Given the continuous nature of probabilities, for a particular outcome class, the estimated probabilities for this class are binned (in this case by intervals of size 0.05). For each bin, the fraction of actual occurrences of this class is then calculated. Therefore, perfect calibration occurs when the fraction of occurrences is equal to the estimated probabilities (i.e. along the line $y = x$). Figure 4 provides reliability

|	BS	BS_{ref}	BSS	Epoch Time (s)
$K = 1$	0.46	0.61	0.25	2180
$K = 2$	0.36	0.49	**0.27**	2929
$K = 3$	0.31	0.40	0.23	3552
$K = 4$	0.27	0.34	0.21	4200

Table 2: *DeepHoops* outperforms the climatology (baseline) model in all cases. Performance is best for $K = 2$ (among the values examined). Epoch Time is lowest over all epochs.

curves for the predicted probabilities of **DeepHoops**. Overall, **DeepHoops** is shown to be well-calibrated, since for all the classes, the reliability curve is close to the $y = x$ line. The drop observed for the "Turnover", is due to the last bin including one data point only.
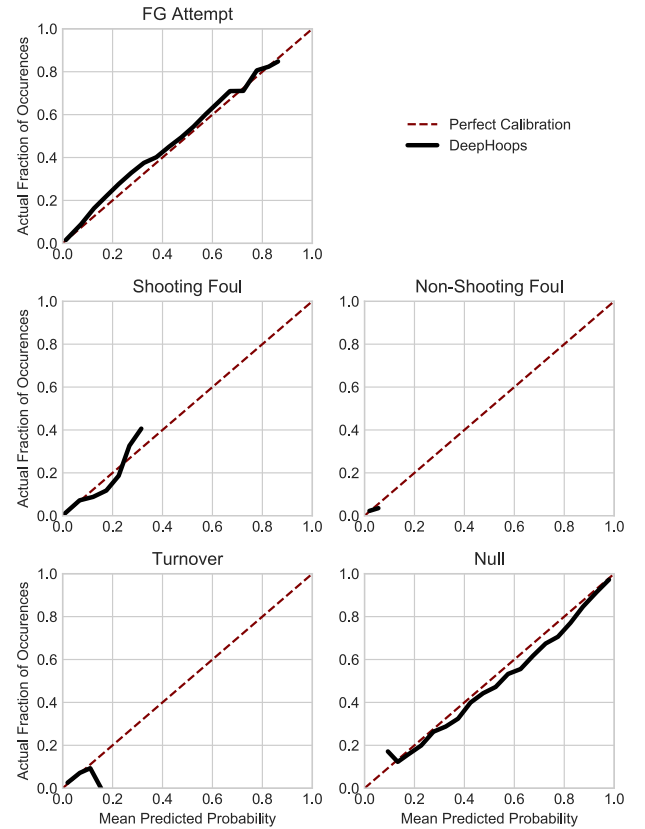

Figure 4: Reliability Curves for *DeepHoops*' probability estimates. The dashed line represents perfect calibration. *DeepHoops* follows this line closely, estimating accurately the outcome probabilities. The tail in the turnover class is due to a bin containing only 1 estimate.

Player Embedding: For the best performing model ($K = 2$) we also trained a version of **DeepHoops** without the player embedding. This model achieves a BSS of 0.26. Thus, while the player embedding improves the performance of the model, this improvement is marginal at best. One explanation may be a difficulty for the network to efficiently learn meaningful representations of players. While designing an alternative neural architecture may allow better
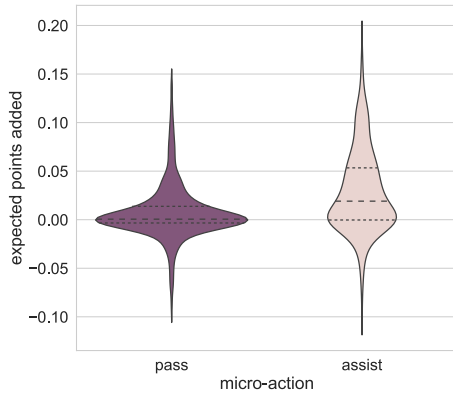
Figure 5: Violin Plot (displays rotated kernel density estimation) of expected points added for randomly sampled passes.
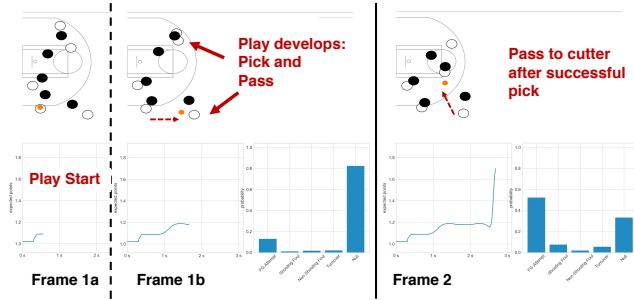
Figure 6: Snapshots of play with running value of expected points and terminal-action probability estimates at each moment. Frame 1a shows the initial moments of the play. Frame 1b shows the play developing: Klay Thompson receives a screen and Patrick McCaw receives the ball; *Deep-Hoops* slightly increases probability of a field foal attempt. Frame 2 shows Thompson receiving the ball just before taking a shot; *DeepHoops* greatly increases probability of a field goal attempt, hence the expected points increases.

propagation of information to remedy this difficulty, another alternative would be to manually encode pertinent information through a pre-trained, external to the network, player embedding. For example, one can build a player embedding based on players' shotcharts that captures the locations from which they take shots similar to Miller *et al.* [23]. This can then be appended to the LSTM's input directly. We plan to explore alternatives such as this for the player embedding further as part of our future work.

4.2 Evaluating Micro-Actions

Evaluating Passes: What is the difference in value between a standard pass and one which leads directly to a scoring event (assist)? Using *DeepHoops* we can answer this type of questions. In particular, we compute the expected points added (see Equation 3) for 512 randomly sampled passes and 512 randomly sampled assists (that is passes that lead to an immediately made FG). We take $\varepsilon = 21$ which corresponds to a 5 moment (fifth of a second) shift around the point of interest (when accounting for the blind-spot). Figure 5

displays a violin plot and quartiles for the expected points added among the different types of passes. Assists are particularly important in basketball and are an indicator of good ball movement and team work. Hence, we would expect *DeepHoops* to value passes leading to a scoring event more highly then those that do not (e.g., a pass in the backcourt). This is clearly the case as we can see by the distributions given in Figure 5. Over 72% of assists increase the expected points. In contrast, standard passes have high density near zero. Furthermore, a Kolmogorov-Smirnov test between the two distributions, rejects the null hypothesis that the two distributions are equal at the 0.1% level.

Realtime Application: Figures 6 and 7 demonstrate snapshots of a realtime application of *DeepHoops* with videos available at https://github.com/anthonysicilia/DeepHoopsRealtimeApplication.

In Figure 6, Klay Thompson cuts down the lane after receiving a screen. Patrick McCaw passes to Thompson and he makes a turnaround fade away jump shot from 19 feet out. *DeepHoops* recognizes the play beginning to develop as the probability estimates of the terminating action are shifted away from **null** and toward field goal attempt, causing a slight increase in the expected points (Frame 1a to 1b). Then, after a successful screen, with Thompson cutting toward the left elbow, *DeepHoops* substantially increases the probability of a field goal attempt, consequently estimating a much higher expected points value (Frame 2).

In Figure 7, Kevin Durant drives towards the basket, but then dishes the ball to Ian Clark who makes a three point jump shot from 23 feet out. This play demonstrates *DeepHoops*' ability to identify when a shooting foul is likely and when it is not. As Durant approaches a group of defenders in the key, the likelihood of both a field goal attempt and a shooting foul increase, while the likelihood of **null** goes down (Frame 1 to Frame 2a). During Durant's pass to an open Ian Clark on the left wing, the probability of a shooting foul drops significantly, as the ball is in flight, but the likelihood of a shot remains relatively stagnant (Frame 2b). When Clark receives the ball moments later, the probability of a field goal attempt increases substantially and the probability of a shooting foul remains low as he has no nearby defenders (Frame 3).

5 CONCLUSIONS AND DISCUSSION

In this paper we introduce a deep learning framework, *DeepHoops*, that is able to track the expected points to be scored by the offense during a possession in a basketball game. *DeepHoops* takes into consideration the players that are on the court during the possession as well as their spatio-temporal distribution. Our evaluations indicate that *DeepHoops* exhibits well calibrated estimates for the probability distribution of the possession outcomes. We further showcase how *DeepHoops* can be used to evaluate micro-actions that have traditionally been challenging to evaluate (e.g., how a standard pass differs from an assist).

In our future work we plan on *customizing* the definition of function ν (Figure 2). Currently we have used league-average statistics. However, Klay Thompson is a better than average shooter and hence, the expected points in Frame 2 at Figure 6 should be higher than what expected from league-average. We can achieve this adjustment by considering a shot quality model (i.e., what is the probability of the FG attempt being successful given the shooter,
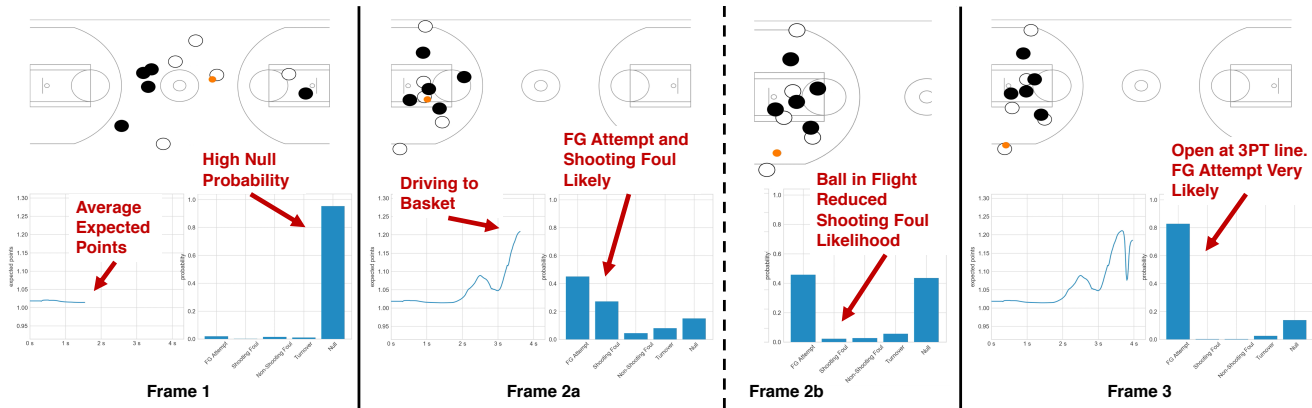
Figure 7: Snapshots of play with running value of expected points and terminal-action probability estimates at each moment. Frame 1a shows average expected points (ball starts in backcourt for most plays). Frame 2a shows Durant surrounded by defenders in the key; *DeepHoops* increases likelihood of both field goal attempt and shooting foul. Frame 2b shows likelihood of shooting foul dropping as the ball is in flight. Frame 3 shows Ian Clark open in the corner, receiving the ball; *DeepHoops* maintains low likelihood of shooting foul, but increases likelihood of a shot attempt.

the distance to the basket etc.). Furthermore, as alluded to above, we are going to explore different approaches to player embedding. Additionally, we will also explore the application of this deep learning framework to estimating the expected points during play in other sports (and in particular American football).

Acknowledgements: We would like to thank Dan Cervone for his valuable comments on our work.

REFERENCES

[1] Alina Bialkowski, Patrick Lucey, Peter Carr, Yisong Yue, Sridha Sridharan, and Iain Matthews. 2014. Large-scale analysis of soccer matches using spatiotemporal tracking data. In *IEEE ICDM*.
[2] Glenn W Brier. 1950. Verification of forecasts expressed in terms of probability. *Monthey Weather Review* 78, 1 (1950), 1–3.
[3] Dan Cervone, Luke Bornn, and Kirk Goldsberry. 2016. NBA Court Realty. In *10th MIT Sloan Sports Analytics Conference*.
[4] Daniel Cervone, Alex D'Amour, Luke Bornn, and Kirk Goldsberry. 2016. A multiresolution stochastic process model for predicting basketball possession outcomes. *J. Amer. Statist. Assoc.* 111, 514 (2016), 585–599.
[5] Nitesh V Chawla. 2009. Data mining for imbalanced datasets: An overview. In *Data mining and knowledge discovery handbook*. Springer, 875–886.
[6] François Chollet et al. 2015. Keras. https://keras.io.
[7] Daniel Daly-Grafstein and Luke Bornn. 2018. Rao-Blackwellizing Field Goal Percentage. *arXiv preprint arXiv:1808.04871* (2018).
[8] Alexander D'Amour, Daniel Cervone, Luke Bornn, and Kirk Goldsberry. 2015. Move or Die: How Ball Movement Creates Open Shots in the NBA.
[9] Alexander Franks, Andrew Miller, Luke Bornn, and Kirk Goldsberry. 2015. Counterpoints: Advanced defensive metrics for nba basketball. *9th Annual MIT Sloan Sports Analytics Conference*.
[10] Alexander Franks, Andrew Miller, Luke Bornn, Kirk Goldsberry, et al. 2015. Characterizing the spatial structure of defensive skill in professional basketball. *The Annals of Applied Statistics* 9, 1 (2015), 94–121.
[11] Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*.
[12] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with LSTM. (1999).
[13] Tilmann Gneiting and Adrian E Raftery. 2007. Strictly proper scoring rules, prediction, and estimation. *J. Amer. Statist. Assoc.* 102, 477 (2007), 359–378.
[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.
[15] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 6645–6649.
[16] Mark Harmon, Patrick Lucey, and Diego Klabjan. 2016. Predicting Shot Making in Basketball Learnt from Adversarial Multiagent Trajectories. *arXiv preprint*

arXiv:1609.04849 (2016).
[17] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[18] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).
[19] Hoang Minh Le, Yisong Yue, Peter A Carr, and Patrick Lucey. 2017. Coordinated Multi-Agent Imitation Learning. In *Proceedings of the 34th International Conference on International Conference on Machine Learning (ICML)*.
[20] Zach Lowe. 2013. Lights, Cameras, Revolution. Retrieved January 24, 2018 from http://grantland.com/features/the-toronto-raptors-sportvu-cameras-nba-analytical-revolution/
[21] Patrick Lucey, Alina Bialkowski, Mathew Monfort, Peter Carr, and Iain Matthews. 2016. quality vs quantity: Improved shot prediction in soccer using strategic features from spatiotemporal data. In *8th MIT Sloan Sports Analytics Conference*.
[22] N. Mehrasa, Y. Zhong, F. Tung, L. Bornn, and G. More. 2018. Deep Learning of Player Trajectory Representations for Team Activity Analysis. In *11th MIT Sloan Sports Analytics Conference*.
[23] Andrew Miller, Luke Bornn, Ryan Adams, and Kirk Goldsberry. 2014. Factorized Point Process Intensities: A Spatial Analysis of Professional. In *ICML*.
[24] Andrew C Miller and Luke Bornn. 2017. Possession sketches: Mapping nba strategies. In *11th MIT Sloan Sports Analytics Conference*.
[25] Allan H Murphy. 1973. Hedging and skill scores for probability forecasts. *Journal of Applied Meteorology* 12, 1 (1973), 215–223.
[26] Evangelos Papalexakis and Konstantinos Pelechrinis. 2018. tHoops: A Multi-Aspect Analytical Framework for Spatio-Temporal Basketball Data. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18)*. ACM, 10.
[27] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026* (2013).
[28] Paul Power, Hector Ruiz, Xinyu Wei, and Patrick Lucey. 2017. Not All Passes Are Created Equal: Objectively Measuring the Risk and Reward of Passes in Soccer from Tracking Data. In *ACM SIGKDD (KDD '17)*.
[29] Lutz Prechelt. 1998. Early stopping-but when? In *Neural Networks: Tricks of the trade*. Springer, 55–69.
[30] Thomas Seidl, Aditya Cherukumudi, Andrew Hartnett, Peter Carr, and Patrick Lucey. 2018. Bhostgusters: Realtime Interactive Play Sketching with Synthesed NBA Defenses. (2018).
[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
[32] Kuan-Chieh Wang and Richard Zemel. 2016. Classifying nba offensive plays using neural networks. In *Proceedings of MIT Sloan Sports Analytics Conference*.
[33] Yisong Yue, Patrick Lucey, Peter Carr, Alina Bialkowski, and Ian Matthews. 2014. Learning fine-grained spatial models for dynamic sports play prediction. In *ICDM*.
[34] Y. Zhong, B. Xu, G. Zhou, L. Bornn, and G. Mori. 2018. Move or Die: How Ball Movement Creates Open Shots in the NBA.