ReLiable: Offline Reinforcement Learning for Tactical Strategies in Professional Basketball Games

Xiusi Chen
Department of Computer Science,
University of California, Los Angeles
Los Angeles, California, USA
xchen@cs.ucla.edu

Jyun-Yu Jiang*
Amazon Search
Palo Alto, California, USA
jyunyu@amazon.com

Kun Jin
Department of Electrical and
Computer Engineering, University of
Michigan, Ann Arbor
Ann Arbor, Michigan, USA
kunj@umich.edu

Yichao Zhou
Department of Computer Science,
University of California, Los Angeles
Los Angeles, California, USA
yz@cs.ucla.edu

Mingyan Liu
Department of Electrical and
Computer Engineering, University of
Michigan, Ann Arbor
Ann Arbor, Michigan, USA
mingyan@umich.edu

P. Jeffrey Brantingham
Department of Anthropology,
University of California, Los Angeles
Los Angeles, California, USA
weiwang@cs.ucla.edu

Wei Wang
Department of Computer Science,
University of California, Los Angeles
Los Angeles, California, USA
weiwang@cs.ucla.edu

ABSTRACT

Professional basketball provides an intriguing example of a dynamic spatio-temporal game that incorporates both hidden strategy policies and situational decision making. During a game, the coaches and players are assumed to follow a general game plan, but players are also forced to make spur-of-the-moment decisions based on immediate conditions on the court. However, because it is challenging to process heterogeneous signals on the court and the space of potential actions and outcomes is massive, it is hard for players to find an optimal strategy on the fly given a short amount of time to observe conditions and take action. In this work, we present ReLiable (ReinforcemEnt Learning In bAsketBaLl gamEs). Specifically, we investigate the possibility of using reinforcement learning (RL) to guide player decisions. We train an offline deep Q-network (DQN) on historical National Basketball Association (NBA) game data from 2015-2016. The data include play-by-play and player movement sensor data. We apply our trained agent to games that it has not seen. Our method is able to propose potentially smarter tactical strategies, compared with replay gameplay data, producing expected final game scores comparable to elite NBA teams. Our approach can be useful for learning strategy policies from other game-like domains characterized by competing groups and sequential spatio-temporal event data.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9236-5/22/10.
https://doi.org/10.1145/3511808.3557105

CCS CONCEPTS

• **Computing methodologies → Planning for deterministic actions**; **Search with partial observations**; **Partially-observable Markov decision processes**.

KEYWORDS

Reinforcement Learning, Policy Learning, Sports

ACM Reference Format:
Xiusi Chen, Jyun-Yu Jiang, Kun Jin, Yichao Zhou, Mingyan Liu, Jeffrey Brantingham, Wei Wang. 2022. ReLiable: Offline Reinforcement Learning for Tactical Strategies in Professional Basketball Games. In *Proceedings of the 31st ACM Int'l Conference on Information and Knowledge Management (CIKM '22), Oct. 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557105

1 INTRODUCTION

An ultimate goal of both data mining and machine learning is to help humans acquire knowledge from environments and make decisions to achieve successful results. In most circumstances, one needs to make sequential decisions to achieve some intermediate goals, which consequently lead to some ultimate goals. However, decision making is typically very hard in complex environments since actors must constantly adapt to dynamic conditions and take actions that they may only hope will yield the largest rewards. Sequential decision making in dynamic environments has drawn the attention of researchers from various areas, such as robotics [26, 41], healthcare [15, 53], and traffic and transportation studies [1, 16].

Professional basketball provides a good example of coaches and players making split-second decisions in response to dynamic environmental factors. Relevant environmental factors include the

*This work was done prior to joining Amazon.

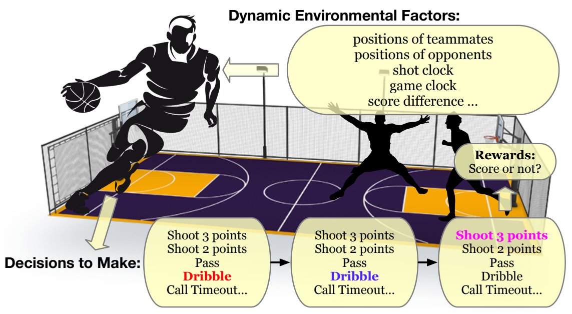


Figure 1: Basketball games are instances of sequential decision making in a dynamic environment. Players and coaches have to make instant decisions that benefit them the best, while external factors such as opponents, game points are changing over time.

current positions of teammates and opponents, the shot clock and game clock, and the score difference, all of which are constantly changing. Furthermore, these factors are heterogeneous in nature under most circumstances. To account for all the critical factors to make a comprehensive decision, one has to first learn representations of these factors to harmonize this **heterogeneity**. These conditions make the optimal sequential decision making a very challenging task. At the elite level, small differences in performance can lead to resounding success or abject failure [18]. Recognizing this, basketball teams usually hire professional staff, including video coordinators and data analysts, who study replay data with the purpose of improving strategic and tactical decision making [40]. However, it is challenging to precisely evaluate the **long-term impact** of sequential atomic actions on the final game outcomes. For instance, even if a team scores at the end of a given possession, it is not necessarily true that every sequential action taken during that possession was optimal. Better strategies may in fact result in higher scores overall. Similarly, if a team does not score, it may be wrong to conclude that all of the decisions taken during possession were bad. "Bad luck" (i.e., stochasticity) may be to blame, instead of deployed tactics. More realistically, any given set of sequential actions taken may be a mix of a few optimal and many sub-optimal decisions, each made on the fly. It is challenging for players to pay perfect attention to all of the environmental signals that would be essential for making perfectly optimal decisions. We therefore explore whether machine learning techniques used to digest massive environmental signals can be used to learn game strategies and potentially help coaches and players make more optimal decisions.

Past studies of basketball [35, 46, 47, 52] incorporate recurrent neural networks to process player-tracking data. The goal is to recognize offensive tactics and to predict the movement of players. These approaches cannot discover optimal sequences of decisions because they lack labeled interactions between the learning agent and the environment. Some recent efforts [38, 46, 51] deploy RL to leverage the game/possession results as rewards to give positive or negative feedback for the strategy learning process. However, these approaches assume the presence of an online environment that can be interacted with to collect a real-time reward, which is often unrealistic since it would be impossible to find an opponent to specifically help one team improve their strategies.

To tackle the above challenges, we formulate each basketball game as a Partially Observable Markov Decision Process (POMDP) where the system dynamics are determined by a Markov Decision Process (MDP), but the agent can only observe *imperfect* signals indicating the likelihood of the system state. In an MDP, actions are determined only by the current state of the system. A POMDP relaxes this Markov constraint in that the best action under the current state can be dependent on historical states. To encode the observation of states, we utilize convolutional neural networks and Transformer [50] to incorporate a continuous sliding window of game state snapshots. The convolutional layers convert the visual signals into low-dimensional vectors, and the Transformer encoder integrates all the snapshots into a vector of hidden state, where all the recent observations are taken into account as well.

To digest the heterogeneous data, we present a well designed pipeline that does multi-modal data representation learning when the offline RL objective is achieved. This capability of incorporating multi-modal signals equips our model with the power of making comprehensive decisions accounting for as many factors as possible.

Another characteristic of our framework is that we train our model under a fully offline setting, where no direct interaction with a real-world environment is available. We must take full advantage of the replay data of previous games. We apply a highly robust mechanism, double Q-learning [49], to improve the Q-value estimation under this offline setting.

To summarize, our contributions are four-fold: (1) We formulate the problem of learning tactics in basketball games as one of solving a POMDP; (2) We propose a framework, ReLiable, to apply offline reinforcement learning techniques to solve the POMDP; (3) A representation learning pipeline facilitates ingesting multi-modal data and models at fine granularity; (4) Extensive experiments to showcase that ReLiable can effectively learn strategic decisions from replay data without interacting with a real environment.

2 RELATED WORK

Applications of Reinforcement Learning. Reinforcement learning is a paradigm that implements learning-based control. Reinforcement learning algorithms have had remarkable success in various application domains, such as robotics [24], self-driving cars [5], industrial control [14], financial markets [36], healthcare [53], news recommendation [54], gaming [44], and advertising [22]. However, many applications of reinforcement learning rely on an online environment that supports interactions. This is a luxury in many settings either because it is costly, unethical, or dangerous to collect data online. As such, it is desirable to learn effective behavior strategies while using only previously generated data. Offline reinforcement learning has been proposed to fully exploit previously collected data without requiring interaction with the environment [2, 11, 13, 28, 32]. Applications of offline reinforcement learning have emerged in domains such as dialogue systems [19], robotic manipulation behaviors [25], and navigation [23].

Sports & Machine Learning. In the field of sports analytics, machine learning and AI has been harnessed only recently for understanding and advising human decisions [48]. Robberechts et al. [42] introduced an in-game win probability model for soccer. Merhej [37] used deep learning techniques to define a novel metric that values defensive actions. Luo et al. [34] proposed a player ranking method that combines inverse RL and Q-learning. Decroos et al. [10] proposed an approach to find patterns in professional soccer as tactics and went on to propose a framework to evaluate any type of player action based on its impact on the game outcome [9]. Sun et al. [45] addressed the trade-off between accuracy and transparency for deep learning applied to sports analytics by proposing a new technique called mimic learning. [43] showcased the utility of statistical analysis (i.e., expected goal value, strategy-plots and passing quality measures) to predict future performance. In [3], a probabilistic graphical model was proposed disentangle the sports teams' luck and skills, and found that luck is as important as skills.

Although there may be no optimal way of playing basketball, there has been research on optimizing certain player decisions within the game. Wang et al. [51] discussed how to leverage reinforcement learning to make better decisions on whether (and when) the defensive team should "double team", a special strategy in which two players closely guard one player from the opposite team to neutralize that offensive player. Neiman et al. [38] focused on the effect of recent field goal attempts on the rate of subsequent shot attempts using a one-state Q-learning model, stating such learning is not guaranteed to improve performance, unless a comprehensive statistical model of the dynamics of the game is present. Liu et al. [33] developed a method that utilized motion capture data to learn robust basketball dribbling moves. By training on both locomotion control and arm control, they were able to achieve robust dribbling under a variety of scenarios. Jia et al. [20, 21] developed a basketball gaming platform called *Fever Basketball* that let developers test their reinforcement learning-based algorithms under their specific video game setting. Based on *Fever Basketball*, Tang et al. further proposed a reinforcement learning method to produce a defensive strategy. These works rely on homogeneous data, and applies ML techniques to conduct analysis on certain aspects of the game. However, they do not focus on utilizing comprehensive multi-modal game information to learn effective team-level offensive tactics while only game replay data is available during training. Imitation learning is another machine learning paradigm that has applications in sports. It implements learning-based control, but requires human expert demonstrations. Le et al. [30, 31] applied imitation learning to explore decision improvement by comparing the specific opponent and "league average", one team might be able to make subtle adjustments to their strategy. Successor studies include extending the action space into a hierarchy [29]. In contrast to reinforcement learning, imitation learning presumes that training examples represent "good" behavior, which is usually impractical.

3 PRELIMINARIES

In this section, we introduce key concepts in reinforement learning (RL) and our notation. We then formally define the problem of reinforcement learning of tactical strategies in basketball games.

3.1 Notation and background

In the RL setting, we usually suppose that an agent repeatedly interacts with the environment. For an agent, the environment provides feedback, such as the next state of the environment and the instant reward for actions taken. This interaction process can be naturally modeled as a **Markov Decision Process (MDP)** [6]. An MDP can be represented as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, r, \gamma)$, where $\mathcal{S}$ represents the set of possible *states* $s \in \mathcal{S}$, $\mathcal{A}$ is the set of all possible actions $a \in \mathcal{A}$ that the agent can take. T defines a state transition mechanism in response to environmental dynamics, which is usually expressed as a conditional probability distribution. Specifically, $T(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$. $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ defines the reward function based on a given state and the action chosen by the agent. Finally, $\gamma \in [0, 1)$ is a discounting factor associated with the learning process that balances between the reward in the current state and the reward in future states.

In a standard setting, the agent (1) starts with an observed state s_t, (2) picks some action $a_t \in \mathcal{A}$ with the potential of maximizing the accumulative reward, (3) enters a new state s_{t+1}, and (4) perceives an instantaneous reward r_t. This process repeats until a terminal state is reached.

The MDP process gives us a good picture of how AI researchers model relations between intelligent agents and the environment. However, it sometimes becomes hard for the agent to fully observe the current state from the environment, so a new paradigm is needed to generalize the regular MDP. To this end, we focus on **Partially Observable Markov Decision Process (POMDP)** models [4].

A POMDP is defined as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, O, T, E, r, \gamma)$ where $\mathcal{S}, \mathcal{A}, T, r$, and γ share the same definitions as in the MDP. O is the set of observations, where each observation $o \in O$ is generated by both the underlying unobservable state and the emission function $E(\mathbf{o}_t \mid \mathbf{s}_t)$.

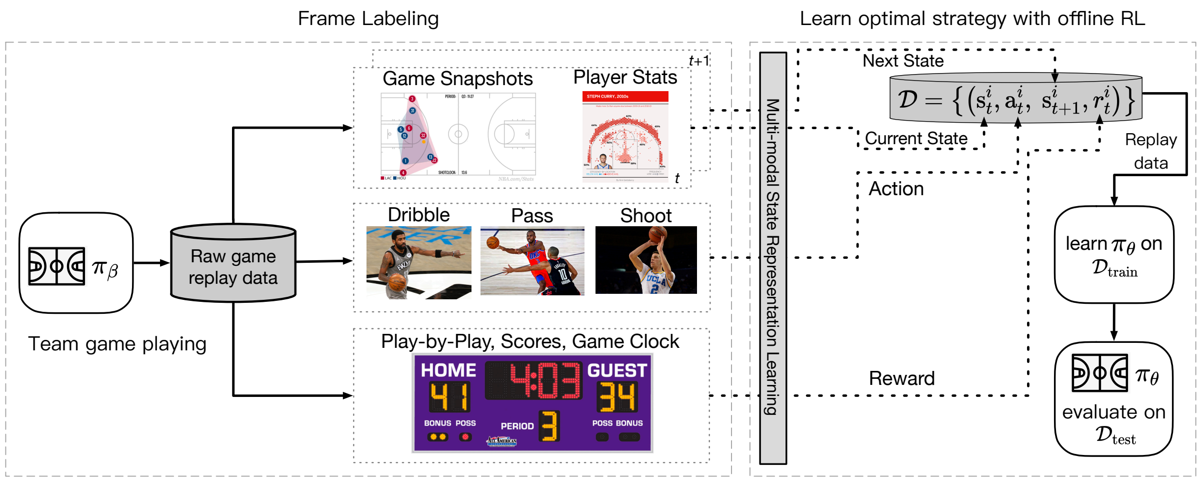
To learn a POMDP, most existing studies focus on online reinforcement learning, and collect rewards and next states from the environment in an on-the-fly manner. The final goal of online reinforcement learning is to learn a policy π that maximizes the cumulative reward $J(\pi) = \mathbb{E}_{\tau \sim p_\pi(\tau)}\left[\sum_{t=0}^{H} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right]$, where the policy π is defined by a distribution over actions conditioned on states $\pi(\mathbf{a}_t \mid \mathbf{s}_t)$ in the MDP setting, or a distribution over actions conditioned on observations $\pi(\mathbf{a}_t \mid \mathbf{o}_t)$ in the POMDP setting. Fundamentally, in MDPs our goal is to find a map from **states to actions**, whereas in POMDPs our goal is to find a map from **probability distributions over states to actions**.

Notice that the reinforcement learning objective $J(\pi)$ is an expectation under a distribution. To fully illustrate this distribution, we have to define a trajectory. The trajectory sequence, or episode is a sequence of states and actions $\tau = (\mathbf{s}_0, \mathbf{a}_0, \ldots, \mathbf{s}_H, \mathbf{a}_H)$. The probability of a trajectory given an MDP and a policy π is given by $p_\pi(\tau) = \prod_{t=0}^{H} \pi(\mathbf{a}_t \mid \mathbf{s}_t) T(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$.

Online reinforcement learning implicitly assumes that the agent has the luxury of directly interacting with the environment. However, this assumption is sometimes impractical. For example, collecting "live" online data may be very expensive, unethical, or dangerous when training an autonomous driving vehicle.

In this paper, we propose the offline reinforcement learning paradigm that operationalizes RL without exploration. It can be regarded as the data-driven version of online RL. The ultimate goal of offline RL is still maximizing the cumulative reward $J(\pi)$, but without the ability to interact with the environment or to collect transitions among states by the policy. In other words, the learning algorithm has to fully exploit the episodes given in a static dataset

$$\mathcal{D} = \left\{\left(\mathbf{s}_t^i, \mathbf{a}_t^i, \mathbf{s}_{t+1}^i, r_t^i\right)\right\} \quad (1)$$



Frame Labeling
Learn optimal strategy with offline RL
Game Snapshots
Player Stats
t+1
t
Dribble
Pass
Shoot
Play-by-Play, Scores, Game Clock
HOME
GUEST
PERIOD
π_β
Raw game replay data
Team game playing
Multi-modal State Representation Learning
Next State
$\mathcal{D} = \{(s_t^i, a_t^i, s_{t+1}^i, r_t^i)\}$
Current State
Action
Reward
Replay data
learn π_θ on $\mathcal{D}_{\text{train}}$
π_θ
evaluate on $\mathcal{D}_{\text{test}}$

Figure 2: Overview framework of ReLiable. The overall pipeline can be split into two major components: Frame Labeling and Offline RL. During Frame Labeling, we annotate each frame with an action. We also append flat features such as game clock, shot clock and player stats to the snapshot to form the episodic data that can be fed into our model. During Offline RL, our model takes both visual features and flat features as input, learns the representation of the multi-modal input, and conducts training and inference.

to produce the best possible policy. Suppose that the episodes in $\mathcal{D}$ are generated by some underlying policy π_β, the actions are then subject to $\mathbf{a} \sim \pi_\beta(\mathbf{a} \mid \mathbf{s})$.

3.2 Problem Formulation

The input of ReLiable is a collection of basketball game logs $\mathcal{D}_{raw}$. The game logs consist of three parts as follows.

Player movement tracking data. The tracking data $\mathcal{D}_{move}$ are static game snapshots that include the positions of all on-court players and the ball at a frequency of 25 frames per second during a game. The progress of any game can be visualized and restored based on the sequence of snapshots.

Play-by-Play data. $\mathcal{D}_{pbp}$ provides a transcript of the game in a format of possessions. It contains 1) the time of the possession, 2) the player who initiated the possession, 3) the outcome of the possession, e.g., how many points are scored, and 4) some other unique identifiers we use to classify the type of possession.

Player stats data. $\mathcal{D}_{stat}$ usually includes player attributes (e.g., height, weight, wingspan, age) and past performance (e.g., minutes per game, points per game, field goal percentage, and 3-point percentage).

For learning purposes, we split $\mathcal{D}_{raw}$ into $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$ as the training set and testing set by the times of gameplays. Formally, we define our task as follows:

Given a collection of game logs $\mathcal{D}_{raw} = \mathcal{D}_{move} \cup \mathcal{D}_{pbp} \cup \mathcal{D}_{stat}$ and an action set $\mathcal{A}$, where each $a \in \mathcal{A}$ is well defined by the discriminative rules on $\mathcal{D}_{raw}$, the task is to assign an appropriate action label a to every frame in $\mathcal{D}_{move}$. In other words, we aim at producing a policy $\pi_\theta(\mathbf{a} \mid \mathbf{o})$ parameterized by θ to output the best action based on the observation related to each frame in $\mathcal{D}_{move}$.

4 METHODOLOGY

Framework Overview Figure 2 illustrates the pipeline of our framework. The raw game replay data is generated by recording the actual games during the 2015-2016 NBA regular season. Each team plays their games according to their policies π_β unknown to our model. The raw game data is multi-modal. Specifically, a game is represented by a sequence of snapshots captured at a high frequency (e.g., 25 frames per second), in which at a given time t, the snapshot contains an image with all player positions and ball position, as well as a set of auxiliary information, such as instant scores, player percentage, shot clock, and game clock, at time t. From the raw game replay data, we extract the game state representation, the action, and the reward on a frame basis to form the replay data $\mathcal{D} = \left\{\left(s_t^i, \mathbf{a}_t^i, s_{t+1}^i, r_t^i\right)\right\}$. Once we derive the replay data $\mathcal{D}$, we feed it into ReLiable to infer effective policy π_θ. Note that during the training process, there is no interaction with any environment. To evaluate the performance of π_θ, we conduct off-policy evaluation on the test set using both action copy and importance sampling. Details on the tasks are discussed in the experiments section.

In summary, our framework employs a dataset $\mathcal{D}$ collected by unknown behavior policy π_β, which can be roughly understood as the "average" policy of all NBA teams. The dataset is collected once and for all, and is not altered during training. The training process is fully dependent on the training set $\mathcal{D}_{\text{train}}$, so it does not interact with environment at all. Once fully trained, we expect π_θ to generalize well on $\mathcal{D}_{\text{test}}$.

4.1 Making Tactical Decisions on-the-fly with RL framework

4.1.1 Game State Representation. The state space for a professional basketball game is extremely large, though presumably not infinite. We want to account for as many game states as possible by considering a continuous state representation that encapsulates player trajectories, player heights, weights, shooting abilities, the shot clock, game clock and current scores for each team. As input to our network, we use both images and flat features.

Our image representation includes three types of channels: i) one court channel encoding the region number of each pixel, ii) 11 trajectory channels (for the 10 players and the ball), and iii) five offensive player shooting percentage channels, each of size 47×50 in the pixel space (i.e., the half court discretized by square feet). When estimating shooting percentages, we use data up to, but not including the current game. This is necessary to respect the causal ordering of events. The result is a 17 channel image. The channels are ordered across images by team and position within a team so that image semantics across examples can be preserved.

To infer the best action at any moment, we see beyond the game snapshot of the current moment. We encode the snapshot sequence of past 3 seconds and take advantage of Transformers which has been proven effective on modeling sequential data, and feed the representation into our model.

Apart from the movement data, other game and player information is also crucial for the agent to make the best possible decisions. Thus, the state representation also incorporates some flat features like the game clock, shot clock, and player historical shooting percentage. For continuous features, we normalize them into range $[0, 1]$, while for categorical features, we use one-hot encoding. Note that for scores, we treat them as categorical features as subtle score difference may have a huge impact on decision making.

Figure 3 illustrates the model architecture of ReLiable. Convolutional networks and Transformer are employed to encode the movement data which are essentially image sequences. Convolutional networks are known for capturing spatial dependencies while Transformers are known for capturing temporal dependencies. Combining them together empowers ReLiable to extract patterns in the features instead of the randomness. To comprehensively incorporate as much game information as we can, the flat features are concatenated with visual features. This concatenation goes through a fully connected module (FC Net) and derives our final state representation.

By leveraging Transformer to capture the temporal dependencies, we are essentially treating the basketball games as a higher-order Markov chains (the order being the sliding window size). Strictly speaking, we do not exactly follow the implementation as a POMDP since typical POMDP uses the belief state to capture everything, but higher-order Markov decision process can be used as a way to tractably perform computation over the PODMP. ReLiable does not compute the posterior over all the past observations, instead we keep a fixed sliding window of observations to approximate the POMDP. From the basketball sense, this also makes sense. When players make decisions on-the-fly, it is the recent game process that matters the most, rather than the whole game procedure.

4.1.2 *Action Space Labeling.*

Our pipeline takes the player movement data as input. To fulfill the POMDP requirement, we need to define the action space and label the player movement data with actions. This paper focuses on the offensive perspective, and therefore in our evaluation, four actions are defined on two sets of experiments: (1) **Shooting 3-points**. Attempting a 3-point shot. If the ball goes in, the offensive team is rewarded 3 points. (2) **Dribble**. The on-ball player keeps dribbling the ball. (3) **Pass**. The on-ball player passes the ball to one of his teammates. (4) **Shoot**. The on-ball player shoots the ball from his current position on the floor.

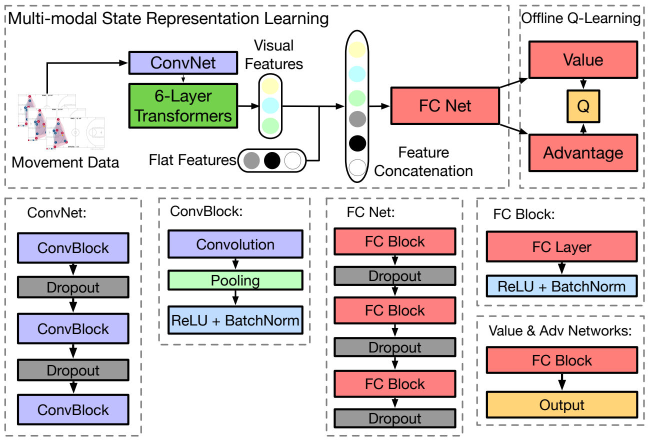


Figure 3: Model architecture of ReLiable. Since we formulate our learning problem as solving a POMDP, past game observations can be incorporated as inputs to predict the best next action. We use convolutional blocks to encode the images into visual features, and exploit Transformer to encode the sequence of game observations. Visual features and flat features are concatenated to form the state representation.

We annotate each and every frame of the raw movement replay data with one of these actions to serve as the ground truth for training and validation. To label a given frame, we inspect a sliding window of snapshots and leverage some simple rules to determine the exact action for each specific frame. The rules that we apply to label actions out of the raw game replay data are as follows. The output of frame labeling tags each frame in the raw data with an action that is being conducted in the frame.

- First, all actions in our action set are highly dependent on the player that currently has possession of the ball. So, our labeling process starts with calculating the distance between the ball and each of the other offensive players.
- We assign the offensive player closest to the ball to be the current ball handler.
- To determine the action of dribbling, we look at a sliding window with a span of 2 seconds. If, during the time window, the ball handler has not changed, then we label all the frames covered by the window as "dribble".
- By contrast, if the ball handler has changed, during this time span, then if the distance between sequential ball handlers is larger than 20 centimeters, we label the frames as "pass".
- Finally, to precisely extract “shooting” frames, we combine the information in the play-by-play data and player movement data. The play-by-play data provides us with all the shooting attempts, so we only have to look at the frames immediately before the attempt. After ruling out those frames determined as "dribble" or "pass", we consider a 5-second sliding window. We label all the frames involved as "shoot" once the height of the ball (relative to the floor) exceeds 10 feet (which is the height of the rim), and the distance of the ball to the rim decreases to near 0.

4.1.3 *Reward.*

The ultimate goal of a basketball team is to score more points than their opponent, thereby winning the game. From the offensive perspective, winning the game is consistent with maximizing the total points scored during a game, which is presumably achieved if a team scores as many points as possible in each and every possession. Therefore, we mainly focus on designing a good reward function based on the points obtained in a possession. According to the basketball rules, points obtained in a single possession can be one of the values in the set: $score \in \{0, 1, 2, 3, 4, 5\}$.

Although obtained points are a good indicator to guide the learning process, we observe that simply rewarding the shooting decision by whether it scores can be misleading, since "bad shots" might nevertheless result in a basket, while "good shots" might miss. A common case to consider is as follows: When it is approaching the end of a game and the score difference is marginal, the trailing team often takes their chances by attempting so-called "low percentage" shots. Sometimes a half-court "buzzer beater" wins the game, but mostly they do not. If we only consider attempts that actually lead to scoring, we might miss these seemingly nonsensical attempts that can potentially help to win the game.

To address this issue, we associate each shot attempt with two regularization terms. First, within each possession, as the shot clock goes to 0, we reward any shot attempt with a value that is a linear function of the shot clock, even if that shot misses. Second, when the game is tight at the end of the fourth quarter, we reward shots that might not seem reasonable, but have the potential to yield a win. Specifically,

$$Reward = score + (24.0 - shot\_clock)/24.0 + game\_clock * NB(5, 2/3)$$

where $score$ denotes the points scored by the offensive team. The second term is the shot clock-based compensation. When a shot happens as the 24 seconds run out, we compensate 1 point for the decision since taking a shot is the only correct action at this point. The third term compensates for potential "buzzer beater" shots. NB denotes the negative binomial distribution with parameters $r = 5, p = 2/3$. This distribution has a probability mass function with a peak at $k = 6$ where k is the support. This means that we empirically assume teams trailing by 6 points are most likely to try the "nonsense" shot attempts. Since the compensation should be a joint distribution of the game clock and score difference, we weigh the score difference with the game clock, so that we compensate more when it is closer to the end of a tight game. We further show analysis that motivates our reward function. During the development of ReLiable, we find there are cases where players seem to make "unreasonable" decisions, e.g., rushing shots or make confusing passes. We further investigated on these "unreasonable" decisions, and find that these cases center around following situations: i) the 24-second shot clock is running out; ii) game clock is running out and the score difference is small. Figure 4 showcases distribution of 3-point attempts over game clock, shotclock and score difference. In Figure 4a, we can see rapid increases in number of 3-point attempts near end of the quarters (every 12 minutes). In Figure 4b, there are quite a lot 3-point attempts made at the end of a possession. In Figure 4c, we can learn that in the case of small score difference, both the leading team and the trailing team tend to attempt 3-pointers to enlarge the advantage or catch up the score. Observing these distributions, we imagine many of these "rushing shots" are in fact wise decision, since these shots can occasionally turn into "buzzer beaters", making the trailing team eliminate the disadvantage to get an overtime. As a result, merely learning from the outcome of these possessions might not reflect the reasonability of these right decisions. To compensate these right decisions, we add the two regularization terms in the reward function.

Table 1: NBA 2015 - 16 Regular Season Game Stats

# Games	# Minutes	# Plays	# Frames
636	30, 528	321, 742	45, 792, 000

4.1.4 Training Process. Figure 3 illustrates the architecture of the training component of ReLiable. We build ReLiable based on the double Q network. We use the double Q network since it has been proven effective in learning defensive strategies [51]. It is flexible to substitute the learning module with other offline RL algorithms such as BCQ [13], REM [2], or CQL [28]. Since we formulate our learning problem as solving a POMDP, previous game observations can be incorporated as inputs to predict the best next action. We use Transformers to encode a sequence of game observation snapshots since Transformers are known for capturing temporal dependencies. We use Convolutional networks to encode image features since we would like to capture the patterns in the snapshot images instead of the randomness. Since the backbone of our framework is a deep Q network, the learning process essentially trains a state-action value (Q-value) estimator that represents a mapping $s \longrightarrow Q^{\pi}(s, a)$ for all actions $a \in \mathcal{A}$. To derive the state representation, we use the convolutional neural network to encode the visual features such as movement snapshots.

Feed forward neural network is used as the function approximator. The function directly approximate the Q-values for each action a_t given a state s_t at timestamp t. Since the action space labeling phase has transformed the raw game data into the episodic training set $\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^{N}$ where each $\mathcal{D}_i = (s_1^i, a_1^i, r_1^i, \dots, s_{T_i-1}^i, a_{T_i-1}^i, r_{T_i-1}^i, s_{T_i}^i)$ is called an **episode** which contains the sequence of states, actions, rewards corresponding to one possession in the game. Our DQN learns its parameters by minimizing the sum of temporal difference (TD) error of all the N episodes in the training set:

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{t=1}^{T_i-1} \left[Q^{\pi^*}\left(s_t^i, a_t^i\right) - \left(r_t^i + V^{\pi^*}\left(s_{t+1}^i\right)\right) \right]^2$$

We represent the derived policy as π^*, and the estimated Q^{π^*} and V^{π^*} represent the Q-value and V-value in general reinforcement learning settings. As a result, our derived policy can be expressed in the following way:

$$\pi^*(a \mid s_t) = \begin{cases} 1, & \text{if a} = \operatorname{argmax}_{a \in \mathcal{A}_t} Q^{\pi^*}(s_t, a) \\ 0, & \text{otherwise} \end{cases}$$

$$V^{\pi^*}(s_{t+1}) = \max_{a \in \mathcal{A}_{t+1}} Q^{\pi^*}(s_{t+1}, a) .$$

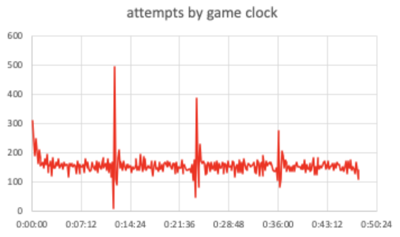
5 EXPERIMENTS

In this section, we present our experimental results in detail. We will first give a thorough description of the data set, then discuss the experimental settings including the input and output of our model and evaluation metrics.

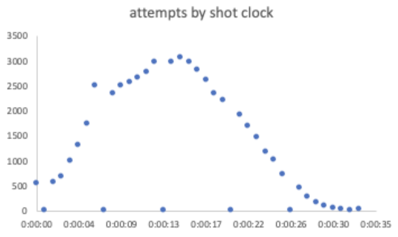
5.1 Dataset

We acquired the data set from a publicly accessible repository [1]. The input data of our model is made up of three parts: (1) **Play-by-Play**:

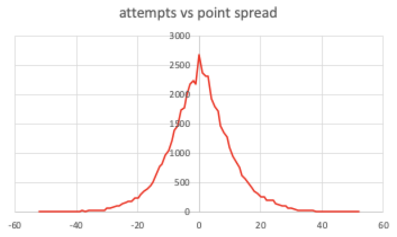
[1]https://github.com/rajshah4/BasketballData/tree/master/2016.NBA.Raw.SportVU.Game.Logs



(a) The correlation between 3-point attempts and game clock.



(b) The correlation between 3-point attempts and shot clock.



(c) The correlation between 3-point attempts and score difference. Negative score difference indicates trailing team behavior.

Figure 4: (a, b, c): The three-point attempts demonstrate some patterns with respect to different factors. Inspired by these patterns, we decide to add regularization terms to the reward function.

Information at the possession level including how the possession ended (jump shot, lay up, foul, etc.), how many points were earned for the offensive team, from which spot the ball shot shot, and the player that shot the ball, etc. (2) **Player movement sensor data**: Court snapshots during games including the positions of players and the ball in Cartesian coordinates. The elapsed time between consecutive frames is 0.04 second. The statistics are listed in Table 1. (3) **Player stats**: The player position and shooting percentage.

5.2 Experimental setting

In general, evaluation of offline reinforcement learning is challenging because we cannot interact with the online environment to collect rewards. Fully exploiting the existing replay data is the only way to perform model validation. To thoroughly examine whether our framework can learn effective strategies, we design two evaluation protocols including *action copy* and *off-policy evaluation via importance sampling*.

In the action copy experiment set, we first focus on the challenge of deciding when to take a 3-point shot. We wish to determine if our model can make good decisions in edge cases such as when to attempt a "buzzer beater" to win the game. Later on, we extend the action space to make it more fine-grained, and investigate whether our model can still choose wisely among dribble, pass and shoot.

In off-policy evaluation, we use importance sampling to evaluate the expected return of our policy. By adding up all expected rewards from each possession for a team, we are able to estimate the point total for a team of agents following the learned strategy policies.

In both experiments, we split the training/testing data chronologically: we use games in 2015 for training, totaling 480 games, and the rest games (in 2016) as the test set, totaling 156 games.

5.3 Baseline Methods

We evaluate the performance of our method against the following supervised and reinforcement learning frameworks:

- **Logistic Regression [39]** is a supervised classification method. It takes flat features as input and computes the correlation between input and labels using a generalized linear function.
- **CNN [27]**, short for convolutional neural network, is a standard image signal encoder classification method.
- **LSTM [17]**, short for Long Short-Term Memory, is a type of Recurrent Neural Network. LSTM networks are well-suited to classifying, processing and making predictions based on time series data.
- **GRU [8]**, short for Gated Recurrent Unit, is another type of Recurrent Neural Network. Compared to LSTM, GRU has a simpler structure, thus achieving higher computational efficiency.
- **Transformer [50]** is a sequence-to-sequence model containing an encoder and a decoder. Here, we only utilize the encoder part. Transformer applies a multi-head self-attention mechanism as an alternative to Recurrent Neural Network, enabling the parallel computation and dramatically enhancing the model efficiency.
- **Soft Actor-Critic (SAC) [12]** is one of the state-of-the-art off-policy actor-critic algorithm. Most RL-based algorithms especially on-policy ones require the interactive environment which is absent in the offline setting. As a result, we pick SAC as our main competitor and adapt it to our offline setting. It expects to learn a policy that acts as randomly as possible while it is still able to succeed at the task. We evaluate SAC under both MDP and POMDP settings.

5.4 Performance Comparison

5.4.1 Action copy. In action copy, we test how likely our model is to behave like the replay data. Different from imitation learning, offline RL does not assume the replay data to be optimal, and may have to handle highly suboptimal data. Offline RL is expected to derive the best policy possible given the data, which means the hope of out-performing the demonstration data.

Simply involving all the trajectories to test similarity seems straightforward, but turns out to be unreasonable since our test data is the raw replay data that includes large numbers of suboptimal plays. As an alternative, we filter out possessions that do not result in scores and focus on how similar our policy behaves to the replay data in the possessions that actually lead to scoring.

Evaluation Metrics. In the 3-point attempts set, whether or not to shoot the ball is a binary decision. We simply use the F1 score as the evaluation metric. In the dribble-pass-shoot set, we use Micro and Macro F1 scores as evaluation metrics.

Table 2 demonstrates the experimental results on the set of 3-point attempts. At any time of the game, every model outputs a result whether or not the better action is to try a 3-point shot at that moment. From the result, we can see that for the binary decision, ReLiable performs better than all the comparative methods. In the set of dribble-pass-shoot, we evaluate similarly as multi-class classification. The results in Table 3 demonstrate a similar pattern as in the previous set, where ReLiable outperforms all the baselines. Combining the two sets, we can see that by considering long-term

Table 2: F1 scores of compared algorithms on 3-point attempts.

Model	F1 score
Logistic Regression	56.28%
CNN	67.10%
LSTM	68.32%
GRU	67.94%
Transformer	70.43%
SAC with MDP	75.27%
SAC with POMDP	78.17%
ReLiable with MDP	76.24%
ReLiable	**81.01%**

Table 3: {Micro, Macro} F1 scores of compared algorithms on {Dribble, Pass, Shoot}.

Model	Micro F1	Macro F1
Logistic Regression	35.17%	27.32%
CNN	42.89%	34.71%
LSTM	45.22%	34.75%
GRU	45.74%	34.14%
Transformer	51.20%	37.48%
SAC with MDP	57.36%	40.43%
SAC with POMDP	70.27%	64.81%
ReLiable with MDP	60.24%	44.09%
ReLiable	**72.95%**	**66.90%**

Table 4: Estimated value of $J(\pi_\theta)$ on test set.

Model	$J(\pi_\theta)$
SAC with MDP	81.36
ReLiable with MDP	94.89
SAC with POMDP (LSTM)	98.42
Seasonal average	102.7
SAC with POMDP (Transformer)	105.75
ReLiable (LSTM)	100.28
ReLiable (Transformer)	**108.16**

rewards, reinforcement learning based methods perform better than supervised learning based methods. In particular, ReLiable makes decisions that are in alignment with those in the successful possessions, and outperforms policy gradient consistently. We attribute this overtake to the double Q-learning architecture, which has demonstrated its superiority in offline reinforcement learning settings. We can also observe that formulating the task as a POMDP by using the Transformer encoder to capture dependencies on recent game snapshots outperforms merely looking at the snapshot of the current timestamp.

5.4.2 Off-Policy Evaluation via Importance Sampling. In the offline RL setting, we have to estimate the cumulative reward $J(\pi_\theta)$ using only the trajectories generated from the unknown underlying policy $\pi_\beta(\tau)$. This idea is also known as off-policy evaluation. In principle, once we can estimate $J(\pi_\theta)$, we can select the policy with the highest cumulative reward. Specifically, we derive an unbiased estimator of $J(\pi_\theta)$ that is dependent on the replay data trajectories:

$$\begin{aligned} J(\pi_\theta) &= \mathbb{E}_{\tau\sim\pi_\beta(\tau)}\left[\frac{\pi_\theta(\tau)}{\pi_\beta(\tau)}\sum_{t=0}^{H}\gamma^t r(\mathbf{s},\mathbf{a})\right] \\ &= \mathbb{E}_{\tau\sim\pi_\beta(\tau)}\left[\left(\prod_{t=0}^{H}\frac{\pi_\theta(\mathbf{a}_t\mid\mathbf{s}_t)}{\pi_\beta(\mathbf{a}_t\mid\mathbf{s}_t)}\right)\sum_{t=0}^{H}\gamma^t r(\mathbf{s},\mathbf{a})\right] \approx \sum_{i=1}^{n} w_H^i \sum_{t=0}^{H}\gamma^t r_t^i \end{aligned}$$

The weight coefficient w_t^i is expressed as $w_t^i = \frac{1}{n}\prod_{t'=0}^{t}\frac{\pi_\theta\left(\mathbf{a}_{t'}^i|\mathbf{s}_{t'}^i\right)}{\pi_\beta\left(\mathbf{a}_{t'}^i|\mathbf{s}_{t'}^i\right)}$ and $\left\{\left(\mathbf{s}_0^i,\mathbf{a}_0^i,r_0^i,\mathbf{s}_1^i,\ldots\right)\right\}_{i=1}^{n}$ corresponds to n trajectories sampled from the underlying unknown tactical strategies from the NBA teams. In our case, these trajectories are game snapshot series from the replay data.

The off-policy method is essentially to see how similar our policy performs to the plays in the test set, with an emphasis on the plays that actually received very high ewards. Essentially, $J(\pi_\theta)$ is a sum of the discounted cumulative reward weighted by the rewards of each step, so if a step in the data actually got high rewards, then we pay more attention to these situations. In order for the weighted sum to be large, we should increase the probability of our policy π_θ taking the action that leads to a high reward in the trajectory generated from the unknown policy π_β.

Table 4 lists the performance of off-policy evaluation. Since off-policy evaluation is only applicable to RL-based methods, we test all relevant methods on all games in the test set, derive the average scores over the games, and exhibit the $J(\pi_\theta)$ value derived from the test set. Essentially, the values in the table stand for the points our policy is expected to score per game on average. We observe that all the RL-based methods can score reasonably high points, among which the ReLiable with Transformer encoder gives the highest points. This indicates that the multi-modal representation learning pipeline indeed helps extract the spatial and temporal dependencies. Taking a closer look, we can notice that ReLiable outperforms policy gradient. We attribute this to the Q-learning backbone, which generally performs better than policy-based algorithms. Generally, turning the problem into a POMDP and leveraging the auxiliary sequence encoder (LSTM or Transformer) to approximate the emission function boost the performance of an RL agent.

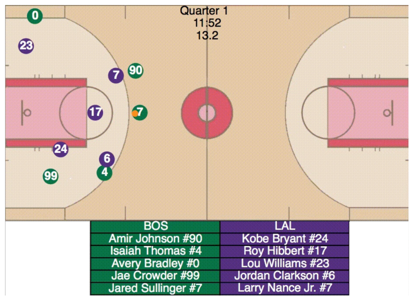
5.5 Ablation study

5.5.1 Partially-Observable Markov Decision Process. Next, we further show the advantage of formulating the task as a POMDP. Compared to MDP, optimizing a POMDP enables RL algorithms to incorporate information beyond one certain state. In our framework, we keep a sliding window of game snapshots at any moment and encode sequences of sliding windows using Transformer [50] to approximate the emission function $E(\mathbf{o}_t \mid \mathbf{s}_t)$ in the POMDP. Tables 2, 3 and 4 show that under different kinds of settings, using a Transformer encoder to incorporate game snapshots from the last few seconds generally outperforms merely looking at a single snapshot by an observable margin. Although the auxiliary information brought by the sequence of snapshots does not strictly follow the Markov property, it equips the model with the capability of making better decisions by considering continuous game processes. A single snapshot only reflects relative positions of the on-court players, the basketball, and the hoops, whereas the sequence of snapshots contains additional information, such as the player handling the ball, whether a shot attempt is a catch-and-shoot (which usually reflects a higher success rate), and player movement speeds. Consequently, feeding sequences of snapshots and encoding them via a Transformer encoder improves the overall performance.

5.5.2 LSTM vs. Transformer. Besides, we compare the performance of using different sequence encoders. From Table 4, we can see that using Transformer as the sequence encoder generally performs better than using its counterpart LSTM regardless of RL algorithms.

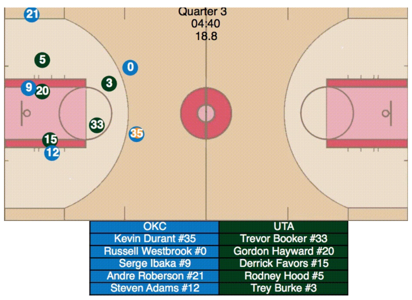
5.6 Case study

We expect offline RL to unearth better policies beyond those present in the replay data. We therefore investigate cases where the decision made by our policy differs from the replay data to provide intuition on why our policy performs well on the test data. Figure 5 illustrates 3 different cases where our policy suggests a different action from the action taken by the teams in the test data. In Figure 5a, Boston player #7, Jared Sullinger, who has an above average 3-point percentage, just caught the ball in an open position. Our model recommends a 3-point attempt, yet Sullinger chose to pass



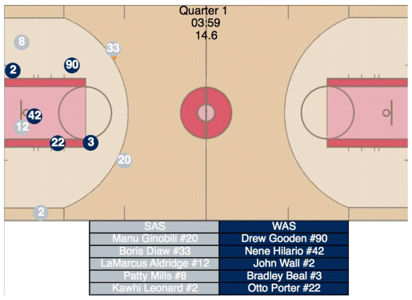
BOS	LAL
Amir Johnson #90	Kobe Bryant #24
Isaiah Thomas #4	Roy Hibbert #17
Avery Bradley #0	Lou Williams #23
Jae Crowder #99	Jordan Clarkson #6
Jared Sullinger #7	Larry Nance Jr. #7

(a) Jered Sullinger gives up an opportunity to shoot a three point.



OKC	UTA
Kevin Durant #35	Trevor Booker #33
Russell Westbrook #0	Gordon Hayward #20
Serge Ibaka #9	Derrick Favors #15
Andre Roberson #21	Rodney Hood #5
Steven Adams #12	Trey Burke #3

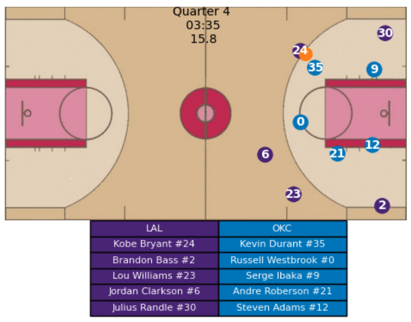
(b) Kevin Durant shoots over a good defender. The shot turns out a miss.



SAS	WAS
Manu Ginobili #20	Drew Gooden #90
Boris Diaw #33	Nene Hilario #42
LaMarcus Aldridge #12	John Wall #2
Patty Mills #8	Bradley Beal #3
Kawhi Leonard #2	Otto Porter #22

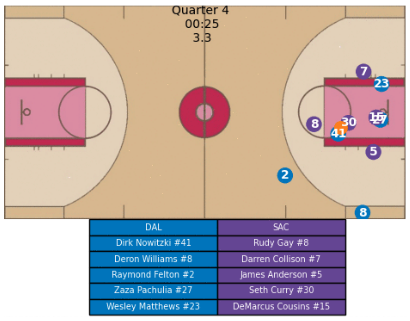
(c) Boris Diaw with good percentage gives up shots, instead dribble and pass.

Figure 5: (a, b, c): Cases that demonstrate "surprises" from our policy. This showcases that the offline RL framework can possibly learn strategies that differ from the existing data that also make sense.



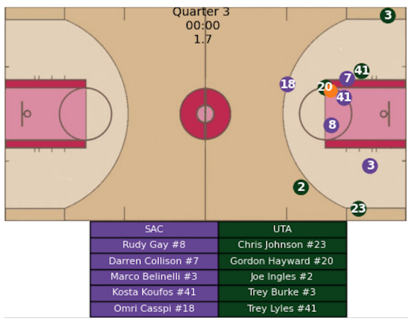
LAL	OKC
Kobe Bryant #24	Kevin Durant #35
Brandon Bass #2	Russell Westbrook #0
Lou Williams #23	Serge Ibaka #9
Jordan Clarkson #6	Andre Roberson #21
Julius Randle #30	Steven Adams #12

(a) Kobe Bryant tries to shoot over Kevin Duran, but misses the shot.



DAL	SAC
Dirk Nowitzki #41	Rudy Gay #8
Deron Williams #8	Darren Collison #7
Raymond Felton #2	James Anderson #5
Zaza Pachulia #27	Seth Curry #30
Wesley Matthews #23	DeMarcus Cousins #15

(b) Dirk Nowitzki shoots over a good defender. It turns out the shot goes in.



SAC	UTA
Rudy Gay #8	Chris Johnson #23
Darren Collison #7	Gordon Hayward #20
Marco Belinelli #3	Joe Ingles #2
Kosta Koufos #41	Trey Burke #3
Omri Casspi #18	Trey Lyles #41

(c) Gordon Hayward makes a buzzer beater at the end of the 3rd quarter.

Figure 6: (a, b, c): Cases that demonstrate "consistence" from our policy. This indicates that ReLiable imitates some critical decisions from the successfully plays.

the ball. The Celtics ended with a missed shot as no better open positions arose later in the possession. In Figure 5b, Oklahoma City (OKC) is in transition from defense to offense. The shot clock has a lot of remaining time. OKC player #35, Kevin Durant, dribbled the ball through the half court. Our model recommends passing the ball, effectively waiting for teammates to set up their positions. Durant took a shot and missed. In Figure 5c, our model suggests Boris Diaw who is a good shooter should take the open shot, while in fact, he gave up the opportunity. The possession ended up without scoring.

Figure 6 demonstrates some cases that discriminates ReLiable from its competitors. In 6a, Kobe Bryant shot over Kevin Durant who is bigger than him. Our model suggests that Kobe should have not taken the shot, while baseline methods suggest the opposite. It turns out that Kobe missed the shot. In 6b, Dirk Nowitzki noticed the shot clock is running out and his team is only leading by 2, so he decided to shoot the ball despite the double team on him. The output of our model is consistent with Dirk's actual behavior, while baseline methods suggest the opposite. In 6c, Gordon Hayward hit a buzzer beater at the end of the 3rd quarter. In the first case, ReLiable successfully avoids a bad shot. In the last two cases, ReLiable is able to figure the player should take the shot in order to hit a buzzer-beater, while comparative methods are not able to take the risk. By looking at these cases, we see that ReLiable is able to discover some decisions even better than the replay data, achieving the goal of offline reinforcement learning.

6 CONCLUSION & FUTURE WORK

In this paper, we propose to study basketball tactical strategy learning with the absence of interaction with the environment. By formulating the basketball games under the context of the partially observable Markov decision process, we are able to apply a data-driven approach based on double Q-learning to derive nearly optimal strategy out of the raw game replay data. We then present ReLiable, an offline reinforcement learning framework for learning effective tactical strategies in basketball games. Experiments and case studies demonstrate the effectiveness of ReLiable over comparative methods, as well as the utility of using Transformer to parameterize the distribution over states under our POMDP setting.

In the future, it is of interest to extend our framework to offline multi-agent reinforcement learning (MARL)[7] setting. Under the MARL setting, decisions are made in a finer granularity since each player should make sequential decisions instead of each team. Compared to single-agent scenarios, multi-agent mode is more challenging since it involves both collaboration and competition. Another direction worth exploring is extending the action space to a hierarchical setting.

ACKNOWLEDGEMENTS

This work was partially supported by the ARO under contract W911NF1810208; NSF 1829071, 2106859, 2119643; NIH R35-HL135772; NIBIB R01-EB027650; Amazon; Cisco; and NEC.

REFERENCES

[1] Baher Abdulhai, Rob Pringle, and Grigoris J Karakoulas. 2003. Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering* 129, 3 (2003), 278–285.

[2] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. 2020. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*. PMLR, 104–114.

[3] Raquel YS Aoki, Renato M Assuncao, and Pedro OS Vaz de Melo. 2017. Luck is hard to beat: The difficulty of sports prediction. In *KDD*. 1367–1376.

[4] Karl Johan Åström. 1965. Optimal control of Markov processes with incomplete state information. *JMAA* 10, 1 (1965), 174–205.

[5] Bharathan Balaji, Sunil Mallya, Sahika Genc, Saurabh Gupta, Leo Dirac, Vineet Khare, Gourav Roy, Tao Sun, Yunzhe Tao, Brian Townsend, et al. 2019. Deepracer: Educational autonomous racing platform for experimentation with sim2real reinforcement learning. *arXiv preprint arXiv:1911.01562* (2019).

[6] Richard Bellman. 1966. Dynamic programming. *Science* 153, 3731 (1966), 34–37.

[7] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. 2010. Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1* (2010), 183–221.

[8] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[9] Tom Decroos, Lotte Bransen, Jan Van Haaren, and Jesse Davis. 2019. Actions speak louder than goals: Valuing player actions in soccer. In *KDD*. 1851–1861.

[10] Tom Decroos, Jan Van Haaren, and Jesse Davis. 2018. Automatic discovery of tactics in spatio-temporal soccer match data. In *KDD*. 223–232.

[11] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. 2020. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219* (2020).

[12] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *ICML*. 1587–1596.

[13] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*. PMLR, 2052–2062.

[14] A Gasparik, C Gamble, and J Gao. 2018. Safety-first ai for autonomous data centre cooling and industrial control. *DeepMind Blog* (2018).

[15] Omer Gottesman, Fredrik Johansson, Matthieu Komorowski, Aldo Faisal, David Sontag, Finale Doshi-Velez, and Leo Anthony Celi. 2019. Guidelines for reinforcement learning in healthcare. *Nature medicine* 25, 1 (2019), 16–18.

[16] Ammar Haydari and Yasin Yilmaz. 2020. Deep reinforcement learning for intelligent transportation systems: A survey. *TITS* (2020).

[17] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[18] Sergio J Ibáñez, Jaime Sampaio, Sebastian Feu, Alberto Lorenzo, Miguel A Gómez, and Enrique Ortega. 2008. Basketball game-related statistics that discriminate between teams' season-long success. *EJSS* 8, 6 (2008), 369–372.

[19] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. 2019. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456* (2019).

[20] Hangtian Jia, Yujing Hu, Yingfeng Chen, Chunxu Ren, Tangjie Lv, Changjie Fan, and Chongjie Zhang. 2020. Fever Basketball: A Complex, Flexible, and Asynchronized Sports Game Environment for Multi-agent Reinforcement Learning. *arXiv preprint arXiv:2012.03204* (2020).

[21] Hangtian Jia, Chunxu Ren, Yujing Hu, Yingfeng Chen, Tangjie Lv, Changjie Fan, Hongyao Tang, and Jianye Hao. 2020. Mastering basketball with deep reinforcement learning: An integrated curriculum training approach. In *AAMAS*. 1872–1874.

[22] Junqi Jin, Chengru Song, Han Li, Kun Gai, Jun Wang, and Weinan Zhang. 2018. Real-time bidding with multi-agent reinforcement learning in display advertising. In *CIKM*. 2193–2201.

[23] Gregory Kahn, Pieter Abbeel, and Sergey Levine. 2021. Badgr: An autonomous self-supervised learning-based navigation system. *IEEE Robotics and Automation Letters* 6, 2 (2021), 1312–1319.

[24] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. 2018. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293* (2018).

[25] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. 2018. Scalable deep reinforcement learning for vision-based robotic manipulation. In *ICRL*. 651–673.

[26] Jens Kober, J Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. *IJRR* 32, 11 (2013), 1238–1274.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *NeurIPS* 25 (2012), 1097–1105.

[28] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems* 33 (2020), 1179–1191.

[29] Hoang Le, Nan Jiang, Alekh Agarwal, Miroslav Dudík, Yisong Yue, and Hal Daumé. 2018. Hierarchical imitation and reinforcement learning. In *ICML*.

[30] Hoang M Le, Peter Carr, Yisong Yue, and Patrick Lucey. 2017. Data-driven ghosting using deep imitation learning. (2017).

[31] Hoang M Le, Yisong Yue, Peter Carr, and Patrick Lucey. 2017. Coordinated multi-agent imitation learning. In *ICML*. 1995–2003.

[32] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643* (2020).

[33] Libin Liu and Jessica Hodgins. 2018. Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *TOG* 37, 4 (2018), 1–14.

[34] Yudong Luo, Oliver Schulte, and Pascal Poupart. 2021. Inverse reinforcement learning for team sports: valuing actions and players. In *IJCAI*. 3356–3363.

[35] Avery McIntyre, Joel Brooks, John Guttag, and Jenna Wiens. 2016. Recognizing and analyzing ball screen defense in the nba. In *Proc. of the MIT Sloan Sports Analytics Conference*. 11–12.

[36] Terry Lingze Meng and Matloob Khushi. 2019. Reinforcement learning in financial markets. *Data* 4, 3 (2019), 110.

[37] Charbel Merhej, Ryan J Beal, Tim Matthews, and Sarvapali Ramchurn. 2021. What Happened Next? Using Deep Learning to Value Defensive Actions in Football Event-Data. In *KDD*. 3394–3403.

[38] Tal Neiman and Yonatan Loewenstein. 2011. Reinforcement learning in professional basketball players. *Nature Comm* 2, 1 (2011), 1–8.

[39] John Ashworth Nelder and Robert WM Wedderburn. 1972. Generalized linear models. *J. R. Stat. Soc.* 135, 3 (1972), 370–384.

[40] Caleb Pagé, Pierre-Michel Bernier, and Maxime Trempe. 2019. Using video simulations and virtual reality to improve decision-making skills in basketball. *Journal of sports sciences* 37, 21 (2019), 2403–2410.

[41] Athanasios S Polydoros and Lazaros Nalpantidis. 2017. Survey of model-based reinforcement learning: Applications on robotics. *JINT* 86, 2 (2017), 153–173.

[42] Pieter Robberechts, Jan Van Haaren, and Jesse Davis. 2021. A Bayesian Approach to In-Game Win Probability in Soccer. In *KDD*. 3512–3521.

[43] Hector Ruiz, Paul Power, Xinyu Wei, and Patrick Lucey. 2017. " The Leicester City Fairytale?" Utilizing New Soccer Analytics Tools to Compare Performance in the 15/16 & 16/17 EPL Seasons. In *KDD*. 1991–2000.

[44] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *Nature* 550, 7676 (2017), 354–359.

[45] Xiangyu Sun, Jack Davis, Oliver Schulte, and Guiliang Liu. 2020. Cracking the black box: Distilling deep sports analytics. In *KDD*. 3154–3162.

[46] Zachary Terner and Alexander Franks. 2020. Modeling player and team performance in basketball. *Annual Review of Statistics and Its Application* 8 (2020).

[47] Changjia Tian, Varuna De Silva, Michael Caine, and Steve Swanson. 2020. Use of machine learning to automate the identification of basketball strategies using whole team player tracking data. *Applied Sciences* 10, 1 (2020), 24.

[48] Karl Tuyls, Shayegan Omidshafiei, Paul Muller, Zhe Wang, Jerome Connor, Daniel Hennes, Ian Graham, William Spearman, Tim Waskett, Dafydd Steel, et al. 2021. Game Plan: What AI can do for Football, and What Football can do for AI. *JAIR* 71 (2021), 41–88.

[49] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *AAAI*, Vol. 30.

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[51] Jiaxuan Wang, Ian Fox, Jonathan Skaza, Nick Linck, Satinder Singh, and Jenna Wiens. 2018. The advantage of doubling: A deep reinforcement learning approach to studying the double team in the NBA. *arXiv preprint arXiv:1803.02940* (2018).

[52] Kuan-Chieh Wang and Richard Zemel. 2016. Classifying NBA offensive plays using neural networks. In *Proc. of MIT Sloan Sports Analytics Conference*, Vol. 4.

[53] Chao Yu, Jiming Liu, and Shamim Nemati. 2019. Reinforcement learning in healthcare: A survey. *arXiv preprint arXiv:1908.08796* (2019).

[54] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A deep reinforcement learning framework for news recommendation. In *WWW*. 167–176.