Jiaxuan Wang*, Fahad Kamran*, Haozhu Wang*, and Jenna Wiens

Exploiting Spatial and Temporal Invariances when Mining Player Tracking Data in Basketball

Abstract: We investigate how domain knowledge can be used to inform modeling decisions, when applying deep learning techniques to spatiotemporal data collected on the court during games in the National Basketball Association (NBA). Multiple different representations for these data have been proposed (*e.g.*, image-based). However, to date, no one has carefully explored the trade-offs among representations and the invariances these representations assume. In this paper, we present a rigorous data-driven analysis of commonly used methods for representing and learning from player and ball trajectory data. We evaluate and compare the utility of flat, image, and time-series input representations. Our results suggest that an image-based representation of these data may be a suboptimal choice for the common task of predicting the outcome of a possession, due to the absence of translation invariance in this task. Though we identify specific invariances that do not hold, we also identify new invariances that, when exploited through data augmentation, lead to improved performance. This work is a necessary first step

towards a better understanding of how to best represent and learn from player tracking data.

Keywords: sports analytics; spatiotemporal data; basketball; data augmentation; invariances; representations

1 Introduction

While advances in deep learning have helped us move away from hand-engineered features, domain knowledge can still help improve model generalization. In particular, domain knowledge can provide valuable insight into what *invariances* are present in the data. The relationship between input and label is said to be invariant to a transformation if the label remains unchanged after applying the transformation. For example, in object recognition tasks using natural images, the labels are often invariant to the location of the object within the image. Knowledge of invariances present in the task of interest can be used to augment the training data [1] or inform model architecture [2, 3, 4].

The invariances underlying natural images have been well exploited in the past. However, the invariances underlying player tracking data have not been extensively studied and are not well known. Player tracking data are spatiotemporal data tracking the two-dimensional location of players during gameplay indexed by time. In recent years, there has been a growing number of researchers looking to leverage these data across sports (including football and basketball) in order to garner insights into what makes for good offensive and defensive strategies [5]. We focus

on data collected in the National Basketball Association (NBA), where locations are tracked at 25 Hz and are precise within a foot.

To address uncertainty around how to best represent these data and lay the foundation for future analyses, we present a careful and rigorous exploration of the advantages and disadvantages of different approaches for leveraging player tracking data in the context of neural networks. We investigate three different data representations based on 1) a flat input, 2) an image-based input and 3) a multivariate time-series input, each used as input to the appropriate deep neural network (DNN) architecture: a feed-forward NN, CNN, and RNN respectively. Each approach makes different assumptions regarding the presence of specific invariances. For example, a CNN applied to an image-based representation of the court assumes that local shifts in player location does not affect the prediction.

We compare the relative utility of each representation in terms of whether or not it improves performance on a given task. Recognizing that some representations might work better for some tasks than others (*e.g.*, identifying a team's overall offensive strategies and classifying a specific play may require different representations), we focus on the task of predicting the outcome of a possession. This task encompasses many other subtasks, since the ultimate goal in a competitive sport is to win a game by winning possessions. The ability to accurately predict the outcome of a possession enables downstream analyses. For example, Wang *et al.* used the outcome of a possession as a reward for learning a strategy for double teaming [5]. Accurate reward estimation facilitates learning such policies. Furthermore, this task has been previously studied in the literature [6, 7, 8].

In addition to comparing the utility of different input representations, we test for the presence of additional domain-specific invariances. Specifically, we are interested in uncovering invariances in player tracking data. We focus on spatial, temporal, and player ordering invariance. For spatial invariance, we hypothesize that randomly flipping the left and right side of the court (with respect to the basket) could help because professional basketball players typically do not exhibit large left and right asymmetries [9]. With respect to temporal invariance, we investigate the length of history needed (*e.g.*, 0.5 second vs. 5 seconds) for accurate predictions. Finally, we explore commonly held assumptions regarding how players should be ordered within a representation [5, 10, 11]. Through a careful evaluation of three common approaches to representing and leveraging player tracking data, we identify key invariances (*e.g.,* invariance to court mirroring, player ordering, and temporal granularity) that could inform and improve future analyses.

We describe our experiments and results below and in addition share all of our code[1] so that others can build off of our work.

2 Background & Related Work

To frame the remainder of the paper, we begin by describing the player optical tracking data. Then, we describe related work on domain knowledge guided input representation and model architecture selection, focusing on applications in sports analytics and specifically basketball.

1 https://tinyurl.com/yxbmzqz6

Player Optimal Tracking Data. Player optical tracking systems have been deployed across sport leagues worldwide [12]. In this work, we consider optical tracking data for basketball collected by SportVu. Briefly, a basketball game involves two competing teams, each consisting of five players on the court with different positions. The game is split into quarters of 12 minutes each, and each quarter can be further split into tens of discrete episodes or "possessions". During each possession, one team is on offense (*i.e.*, has the ball) and has the opportunity to score anywhere from 0 to 4 points, while the other team defends. Possessions last no longer than 24 seconds.

The optical tracking system records the positions (x, y) of all players and the ball at 25 Hz. The data are augmented with game clock and play-by-play event information, so that we know when each possession starts and ends, who is on the court, and the outcome of the possession. These spatiotemporal data represent a rich testbed for hypotheses and opportunities for gaining a data-driven competitive advantage [5, 7, 11, 13, 14, 15, 16].

Related Work. Despite DNNs' capability of extracting useful representations from data automatically, their performance is largely affected by the input representation and selected network architecture. Researchers often refer to domain knowledge when making such modeling choices [3, 4, 17, 18].

In the context of analyzing player tracking data, multiple representations have been explored. Previous work can be roughly divided based on input modality as either i) a flat input, ii) an image-based input or iii) a multivariate time-series input. The first approach summarizes player trajectories and in-game statistics into a single fixed-length feature vector that can be used as input to a fully-connected

neural network. This is one of the simplest representations and has been widely used [13, 19, 20, 21, 22, 23, 24, 25]. However, such an approach requires careful feature engineering, to ensure that relevant statistics are included. For example, McIntyre *et al.* computed pairwise distances between players to capture relative spatial information for classifying defensive strategies [13]. In the second approach, player trajectories are mapped onto an image (or images) of the court. In this setting, time is frequently encoded using some kind of fading when mapping trajectories. This approach preserves spatial information and is one of the most popular ways to represent player tracking data [5, 6, 26]. The third approach encodes each trajectory as a multivariate time series and uses a RNN for prediction. Though not as popular as CNNs, RNNs have been used by researchers analyzing player tracking data, because they naturally handle variable length input and capture temporal dependencies [10, 11, 27, 28, 29]. Aside from the three common approaches, others have used combinations of time-series and image-based representations [10, 11].

Despite a large body of work on using player tracking data, to date, no one has explored the effectiveness of flat/time-series/image representations for predictive modeling using DNNs. Thus, in this paper we compare these three representations on a common prediction task and give practical suggestions on the effective use of player tracking data for sports analytics.

3 Methods

In this section, we begin by introducing notation used throughout. Then, we present details regarding each input representation and the corresponding model

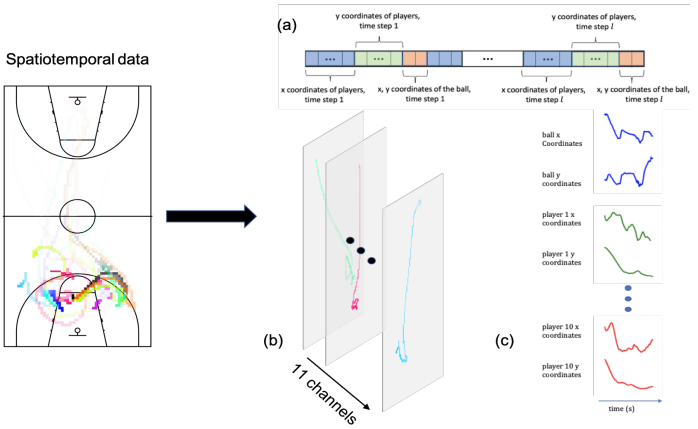

Fig. 1: From spatiotemporal player tracking data (*i.e.*, timestamped player locations on the court), we extract **(a)** a flat representation, **(b)** an image-based representation, and **(c)** a time-series representation. In the flat representation, we concatenate the previous l (x, y) positions of each player on the court along with the ball. These data are then used as input to a feed-forward neural netword. In the image-based representation, each player and the ball is represented by a trajectory faded based on time and used as one of 11 input channels (10 players + the ball). These data are then used as input to a CNN. In the time-series representation, we capture each player's and the ball's (x, y) locations along the temporal dimension as a multivariate time series. These data are then used as input to an LSTM.

architectures. In addition, we describe the invariances we set out to test and conclude with a detailed description of the training and evaluation setup.

3.1 Problem Setup and Notation

We consider a multi-class prediction problem, in which we aim to predict the outcome of a possession given trajectory data. The label set $\mathcal{Y} = \{0, 1, 2, 3, 4\}$ denotes the set of possible outcomes for a possession in terms of points. We assume training data of the following form $\mathcal{X} \subset \bigcup_{t=1}^{t_{\max}} \mathbb{R}^{m \times t \times d}$, where $\mathcal{X}$ is a multivariate trajectory of variable length (up to $t_{\max}$), m is the number of trajectories for a possession, and d is the dimension of the trajectory at each time step. In the game of

basketball, the maximum possession length is 24 seconds and the data are sampled at 25 Hz, we have $t_{\max} = 24 \times 25 = 600$. Meanwhile, $m = 11$, corresponding to the trajectories of 5 offensive players, 5 defensive players, and the ball. $d = 2$ denotes the (x, y) coordinates on the court. Though the z coordinate (*i.e.*, the height of the ball) is available, we do not include it since it did not improve model performance in preliminary experiments.

For a possession $x \in \mathcal{X}$, we use $x_i \in \mathbb{R}^{t \times d}$ as shorthand to denote the i^{th} trajectory, $x_t \in \mathbb{R}^{m \times d}$ to denote the snapshot of possession at time t (we also refer to it as a frame), and $x_{i,t} \in \mathbb{R}^d$ to denote the i^{th} player or ball's location at time t. As a convention, we use capitalized letters such as X and Y to denote the random variables with sample space $\mathcal{X}$ and $\mathcal{Y}$, respectively. Similarly, we use X_t to denote the random frame at time t and X^t to denote the joint random variable of all random frames from time 0 to time t.

3.2 Building Different Representations

In this section, we describe in detail the construction of the three representations under consideration: flat, image, and time series, as well as their corresponding model architectures.

Flat representation. We construct a flat input representation by concatenating the player and ball coordinates of a possession trajectory into a single vector (**Figure 1 (a)**). Similar approaches have been applied in the literature [13, 19, 20, 21, 22, 23, 25]. For example, Kempe *et al.* downsampled trajectory data by averaging the locations within every second, and used these downsampled

trajectories as input to a neural network for classifying plays [19]. Similarly, here, we use this vector as input to a feed-forward fully connected neural network with three hidden layers of size $512 \times 256 \times 128$ for predicting possession outcome. To generate a fixed length input (required by the NN), we consider only the previous l time steps when making a prediction. By concatenating the last l frames of player and ball trajectories in a possession, we generate a 1-dimensional vector in $\mathbb{R}^{m \cdot d \cdot l}$ that can be used as input to our feed forward network. As a default, we use $l = 25$ (*i.e.*, 1 second of data), but explore the effects of different values of l in Section 4. As is common practice, we use batch normalization [30] and ReLU activation.

Image representation. Next, we consider a multi-dimensional (*i.e.*, non-flat) representation that can be used as input to a convolutional neural network (CNN). CNNs have been applied widely to player tracking data. For example, Brooks used an image representation to understand how off-ball players contribute to the success of a possession [26]. Brooks linearly faded the pixel values and focused on half-court actions with a 25×25 sized image. Similarly, Harmon *et al.* discretized the court into 94×50 pixels, tracing player trajectories, while exponentially fading the pixel values to encode the passage of time [6]. Wang *et al.* used a similar representation for learning defensive strategies with deep reinforcement learning [5]. Building on previous work, we create an image representation for each possession, consisting of m channels corresponding to each of the ten players and the ball. Each channel is a 94×50 array with each cell of the array covering one square foot of the court. We also considered a half-court representation, but did not observe any difference in our experiments. For each frame in a specific player's trajectory, we fill in the pixel value in the channel corresponding to the player's position at

that frame. We start with a value of $v_t = 1$ for the entry corresponding to the last frame, and we exponentially decay this value frame by frame such that the most recent information is decayed the least (**Figure 1 (b)**). In Section 4, we investigate the effects of this decay value with a default of 0.99. Thus, the pixel value at the position corresponding to the i-th frame is $v_i = 0.99^{t-i}$. This representation encodes variable length possessions, since all trajectory information is encapsulated in a fixed-size array.

For the prediction task, we use these representations as input to a CNN consisting of three convolutional layers followed by two fully connected layers. We use an architecture that has been used by others leveraging image-based representations of player tracking data [5, 6]. Between each layer, we use ReLU activations followed by batch normalization. For the convolutional layers, before ReLU, we add max pooling with a kernel size of two. Each convolutional layer produces 32 output channels and a kernel size of three.

Time-series representation. Finally, we consider a time-series representation that can be used as input to a RNN. RNNs are useful architectures for sequential inputs (*e.g.* time series) due to their ability to keep internal state throughout the temporal processing of a sequence. For example, Shah and Romijnders trained an RNN on ball trajectories to predict the success of three pointers [29]. Here, we encode the trajectory information for the ball as two signals, one for each of the x and y coordinates. We represent each player's trajectory similarly, according to a two-dimensional time series, resulting in an $m \times d$ dimensional input, where each input dimension is a signal in $\mathbb{R}^t$ and t is the length of the possession (**Figure 1 (c)**). Since each time step (*i.e.*, frame) of a possession corresponds to a

feature vector in the overall representation, the length of this representation will vary depending on the length of a possession. Given that data collected early on in the possession may be crucial to the end result of the possession, we chose a long short-term memory (LSTM) architecture [31] since they effectively capture long-term dependencies. Our LSTM has one layer with a hidden size of 300, followed by two fully connected layers. Between the fully connected layers, we use a ReLU activation function, followed by batch normalization.

In all three representations, we align possessions spatially to ensure that the direction of the offense is always the same by flipping the input appropriately[2].

3.3 Invariances in Representations

Every task may exhibit some form of invariance, *i.e.*, a set of transformations that when applied to $x \in \mathcal{X}$ would not change its corresponding target $y \in \mathcal{Y}$. Knowing what invariances are present allows one to remove artificial variations in data or augment the training data and improve model generalization. In this work, we check for the existence of temporal invariance, spatial invariance, and invariance to player ordering. We explain each type of invariance below.

Temporal invariance. In the context of predicting possession outcomes from player trajectories, we explore whether the performance of the model is invariant to the amount of history used during training. This corresponds to testing to what extent the Markov property holds for our task, *i.e.*, whether Y is independent from X^{t-1} given X_t. We expect that the performance of the model will improve

2 In basketball, the end is arbitrary aside from determining who's on offense/defense.

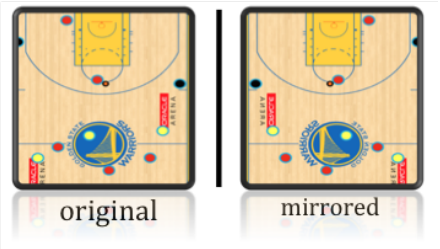

Fig. 2: By mirroring the court, we can identify/exploit any left-right invariance, if present. If the side of the court affects the outcome of the possession, then mirroring the court would destroy this information, leading to worse performance.

as we increase the amount of history used for training the model. In addition, we hypothesize that if what happens early on in a possession greatly affects what happens later, then the RNN will more effectively leverage this history compared to the other methods.

Spatial invariance. In our context, given that professional basketball players do not exhibit large asymmetries in their game (otherwise such weaknesses could be exploited) [9], we hypothesize that the player trajectory is invariant to flipping the court along the axis aligned with the long side. To test this hypothesis, we propose a novel data augmentation scheme in which we randomly flip the left and right side of the court from the perspective of players when training the model. We refer to it as 'mirroring' the court (**Figure 2**). If NBA players have a noticeable difference in attacking from the left versus the right, randomly flipping the court in this way should hurt performance. Otherwise, we expect it will help. In order to test this spatial invariance, we modify our training scheme. For each possession during training, with equal probability, we either modify the input so that the right and left-hand side of the court are flipped or leave the possession unmodified.

We evaluate our trained models on unmodified possessions in the validation and test sets.

Invariance to player ordering. We investigate the effects of player ordering on prediction performance. Player ordering refers to the order in which player trajectory data enter the learning algorithm. For example, in **Figure 1 (b)** the order of the channels corresponds to different players in an image-based representation. Each possession $x \in X$ comes with an arbitrary ordering of players. However, when comparing two possessions, for computational efficiency, generally some ordering is assumed. In the literature, researchers often assume that the choice of ordering affects predictive performance [5, 10, 11]. For example, you might want to compare the trajectories of players with the same positions. Here, we test this assumption by comparing three approaches for ordering players in the model input. The first approach we consider is a position-based ordering; we order players by their basketball positions (*e.g.*, center, point guard, etc.). Some players may play multiple positions; we assign such players their own position type (*i.e.*, create a new position). After creating these new positions, all players could be classified into one of nine unique positions. The second approach corresponds to an ordering based on arbitrarily assigned player ID. These player IDs remain fixed throughout training such that relative ordering between two players is always preserved. We refer to this approach as arbitrary ordering. The final approach is a permutation invariant ordering, in which we repeatedly randomly permute the ordering of the players during training. For all three ordering methods, we group the players according to whether they belong to the offense or defense and apply the aforementioned ordering methods within each group. For the CNN, we

consider an extra permutation invariant ordering by summing players' trajectories. We reduce five offensive player channels into one, five defensive player channels into one, and keep the ball channel intact, resulting in three channels.

While the set of transformations examined here are far from exhaustive, they represent a good starting point, as they are based on commonly held assumptions for player trajectory data.

3.4 Training and Evaluation Details

We consider the task of predicting the outcome of a possession (in terms of points) at every second of a possession using only information available up to that point. In contrast to previous work [6], we do not assume we know when a possession ends. This is a more realistic setup, since information regarding the length of a possession is not available in real-time. Our formulation, however, results in a more challenging prediction task.

We use the term prediction horizon to refer to the gap between the time of prediction and the end of the possession. Note that, though this information is only available retrospectively, it is a good proxy for the difficulty of the task and helps in evaluation. Formally, for a possession x of length t, a prediction horizon of h means we use only data up to time step $t - h$, or x^{t-h} for prediction. We expect that longer prediction horizons correspond to more difficult tasks, since this requires predicting further into the future. As we mentioned above, it is important to learn a model that can generalize to different prediction horizons, since we do not know in advance when a possession will end. However, there exists a trade-off between

the size of the prediction horizon considered during training and the resulting generalization ability. At one extreme, if we use a small prediction horizon across all training examples, our model may have difficulty generalizing to other prediction horizons. At the other extreme, if we train on all possible prediction horizons, model performance may still suffer, since this increases the difficulty of the task. We balance these two extremes, by training on examples with prediction horizons ranging from 0 to 2 seconds. We evaluate across a range of prediction horizons. Based on preliminary results, we noticed that no model was able to generalize past a prediction of horizons 6 seconds. Thus, throughout our experiments we report results for prediction horizons ranging from 0 to 6 seconds.

4 Experiments and Results

In this section, we describe the training task, our experimental setup and our evaluation procedure. Then, we present and analyze results for a series of experiments.

4.1 Training Data

We use NBA player tracking data from three seasons, corresponding to $2014-2015$, $2015-2016$, and $2016-2017$. We split the data into train, validation, and test sets temporally, keeping the most recent season as the held-out test set. Since we are interested in predicting possession outcomes, we focus on trajectories over the course of a possession (as opposed to over the course of a game) and consider only those possessions corresponding to continuous play. In addition, we focus

on possessions where a shot event is recorded in accordance with Brooks and Harmon *et al.* [6, 26]. When calculating the outcome of a possession, we include both field goals and free throws. We exclude rare possessions that result in more than four points and restrict possession length in the range of 12 to 24 seconds. This restriction reduces the effects of confounding due to the length of a possession. After applying these exclusion criteria, our final dataset contains 123,623 training possessions, 31,033 validation possessions, and 78,443 testing possessions. Over all possessions in our study dataset, 57.7% result in a score of 0.

4.2 Experimental Setup & Evaluation

To train each of the networks described in the methods section, we minimize cross-entropy loss. We optimize the parameters of the network using Adam [32]. We use a fixed training budget of 20,000 mini-batches (each of 128 samples), saving the model that led to the best performance on the validation data. We implement all neural network models with PyTorch [33].

While Harmon *et al.* [6] used accuracy to measure validation and performance, we measure validation performance in terms of the area under the receiver operating characteristics curve (AUC), since it is invariant to class imbalance (here, we have more missed shots than made shots)[3]. Given our multi-class classification setup, we evaluate the performance of each representation/model on the held-out test set in terms of the macro-averaged AUC. We measure the macro AUCs over prediction

3 Though optimized for a different validation metric, our best validation image representation model achieves 60% accuracy on test similar to 61% accuracy reported in [6].

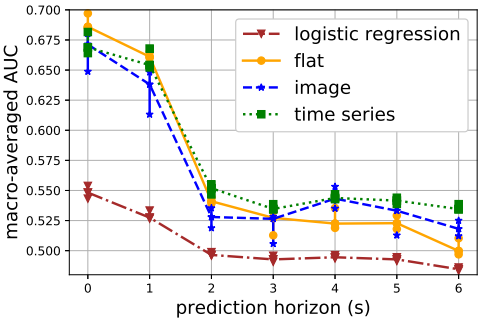

Fig. 3: Comparison of input representations results for all three runs shown with median highlighted. For short-term prediction (*i.e.*, prediction horizon $<$ 1s), the flat input representation performs best. But for more difficult tasks (*i.e.*, longer prediction horizons), the time-series representation performs consistently better. All deep learning approaches outperform the logistic regression baseline.

horizons from 0-6, averaging in increments of 1s. Since the initialization of a network can affect final model performance, we repeat each experiment three times for three different random initialization.

4.3 Relative Utility of Different Input Representations

We ran extensive experiments across all input settings. Here, we focus on a subset of these results. In the interest of transparency and reproducibility, we include results from all experiments in the Appendix. To compare the utility of the different input representations, we chose the best model from each modality using the validation set and report test performance (**Figure 3**). To showcase the utility of deep learning approaches and demonstrate the difficulty of our prediction task, we also include logistic regression with flat input as a baseline. Note that across prediction horizons, logistic regression performs the worst. For a prediction horizon

of 0 seconds (*i.e.*, when a shot goes up), the flat input performs consistently better than both image and time-series representations. **However, for longer prediction horizons, the model based on the time-series representation dominates.** We hypothesize this is due to the fact that RNNs are designed to model important temporal dynamics. For short prediction horizons, the image representation performs the worst on average. Note that the translation invariances commonly exploited by CNNs may not hold for short-term prediction. Specifically, the pooling layers commonly found in CNN architectures may blur the spatial locations necessary for accurate short-term prediction. Moreover, we hypothesize, that the poor performance of the image representation is due, at least in part, to the sparsity of the signal. In such representations, the majority of the input is zero and does not contain any information (*i.e.*, players occupy only a small fraction of the court). This adds unnecessary complexity to the learning task and forces the network to learn what to pay attention to and what to ignore. Experiments using a half-court representation show a slight increase in performance (**Figure 9** in Appendix).

We hypothesize that the poor performance of the flat input representation at larger prediction horizons is due to this representation's inability to adapt to variable length input (recall all possessions are summarized by considering only a fixed window in time). To address this issue, others have looked at hand-engineering features, extracting fixed-length summaries of variable length trajectories [13]. In contrast, the image and time series based representations do not require such hand-engineered features and can easily adapt to variable length inputs. Thus, for the remainder of our experiments we focus on only these two representations.

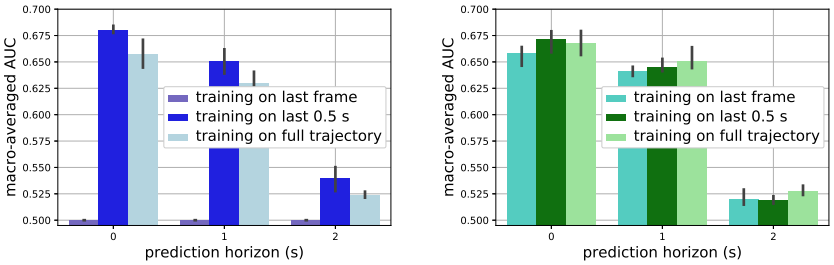

Fig. 4: Testing invariance with respect to length of history. Across prediction horizons a model trained on shorter trajectories (the most recent 0.5 seconds) leads to **(a)** better performance when using images, **(b)** and little or no difference when using time series.

4.4 Exploring Temporal Invariance

In this experiment, we explore temporal invariance. Specifically, we investigate whether the predictive performance of a model depends on the amount of history used during training. We expect that more information (*i.e.*, a longer trajectory history) will result in better predictions. To test this hypothesis, we train models using trajectories of different lengths. We compared the effects of using only the last frame, using only the last 0.5 seconds, and using the full history of the possession (**Figure 4**). Counter to our intuition, models trained on longer trajectories do not perform better than those trained on shorter trajectories. Note that this result is not confounded by possession length, since we restricted our dataset to only possessions of at least 12s. **This suggests that all of our models rely on short-term dependencies within a possession**. Perhaps the effects of a decision made early on in a possession are transient. In such cases, a large portion of the trajectory would have little utility in making the prediction. For the CNN, the model does not learn anything using only a single frame (*i.e.*, snapshot). Again, we hypothesize

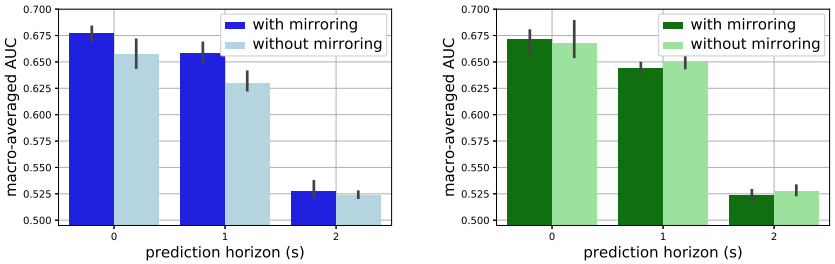

Fig. 5: Testing spatial invariance, by mirroring the court. For the image-based representation (a) mirroring the court consistently helps. For the time-series representation, mirroring the court does not hurt model generalization.

that this is due to the high degree of sparsity in this representation. In contrast, the RNN that uses only the last frame performs nearly identically to one based on the full history. These results suggest that in predicting the outcome of a possession at a specific point in time, data pertaining to the most recent positions on the court are most important (note that unlike in **Figure 3**, here we have not added in player information, which we later show in the Appendix that helps an RNN remember). From these observations, we conclude that in at least some settings, the predictive power of a trajectory is invariant with respect to its length. Going forward, this implies that one may be able to train more efficiently by using shorter trajectory histories.

4.5 Exploring Spatial Invariance

In our next experiment, we explore a type of spatial invariance. As described in the methods section, we randomly flip the left and right side of the court from the perspective of the players facing the basket head on. If such spatial invariance

Tab. 1: Player ordering does not appear to affect performance. AUC@0-2 denotes the average macro AUC evaluated at prediction horizons of 0-2 seconds. AUC@3-6 denotes the average macro AUC evaluated at prediction horizons of 3-6 seconds. We show the median, (min, max) results across runs for the image-based representation. Similar results for time series and flat input are included in the Appendix.

ordering	AUC@0-2	AUC@3-6
arbitrary	0.600 (0.591, 0.606)	0.514 (0.503, 0.517)
position	0.601 (0.598, 0.611)	0.510 (0.508, 0.510)
random permutation	0.601 (0.592, 0.604)	0.508 (0.504, 0.512)
sum trajectory	0.606 (0.585, 0.608)	0.514 (0.498, 0.522)

exists, then randomly flipping the court could exploit this invariance, leading to better generalization. We find that mirroring the court consistently improves the image representation's performance for short time prediction (**Figure 5 (a)**). The boost in performance suggests that mirroring the court helps alleviate the problems associated with the sparsity inherent to image-based representations of the possession. In other representations, mirroring the court does not hurt performance (**Figure 5 (b)**). Thus, we conclude that a left-right spatial invariance holds for the task of predicting NBA possession outcome, using player tracking data. **Overall, mirroring the court appears to be a useful data augmentation technique that could be employed to improve performance on this or possibly other tasks**.

4.6 Exploring Invariance to Player Ordering

To test the importance of player ordering, we consider ordering based on player id (also referred to as arbitrary ordering in plots), ordering based on player position, and ordering with randomly permuted players for each trajectory. Overall, player

ordering, as defined Section 3.3, makes no difference in test performance for all the input modalities (Table 1 and **Figure 6**, **Figure 7**, and **Figure 8** in Appendix). **This suggests that player ordering, despite containing player information, is perhaps too weak of a signal for models to exploit.** Hence, there exists an invariance to the ordering of the players in the input for each representation.

5 Conclusions

We explored the advantages and disadvantages of different representations of spatiotemporal basketball data. Through our analysis, we challenged some of the common assumptions in applying deep learning to player tracking data, leading to the following conclusions.

CNNs struggle with the task of predicting possession outcome. Though image-based representations (with fading) are a natural way to represent spatiotemporal data, they lead to worse performance compared to other representations. This finding is in direct contrast to a lot of previous work that has relied on image-based representations of these data. We hypothesize that the sparsity of the representation contributes to the inability of the CNN to capture relevant information. Moreover, a CNN assumes translation invariance that may not hold for these tasks in which the courts are aligned and the position of the basket never changes.

Trajectory information is most useful in the short term. Models perform similarly when trained on either the full history of the possession or only the

most recent frames. Importantly, trajectory information appears most useful near the end of a possession (*i.e.*, during the last second). Going forward, practitioners may want to consider efficient training schemes that rely on shorter trajectories.

'Mirror' the court to augment your training data. We show there exists a left-right spatial invariance (relative to the basket) when predicting the outcome of a possession. This spatial invariance suggests that NBA players are similarly effective on the left side of the court as they are on the right side. Beyond this task, mirroring the court could prove to be a valuable data augmentation technique more generally.

This work represents a rigorous analysis of common assumptions in applying deep learning to player trajectory data in the NBA. Such analyses are important because they can inform future applications of deep learning in similar situations, fueling progress and advancements. Beyond basketball, we expect the lack of translation invariance to hold in other team sports such as soccer, football, and hockey, since where the players are with respect to each other and the goal matters. Thus, a CNN applied to an image-based representation of player tracking data may not be the appropriate choice in those situations. In addition, we expect that 'mirroring' the court/field/rink may be an effective data augmentation strategy for other sports at the professional level, while less so in amateur games. All of our code is available so that such analyses can be easily replicated on other datasets.

Regardless of the application, it is important to question the use of different input representations and model architectures. Much of the developments in deep learning have been fueled by specific application domains (*e.g.*, natural images) and lessons learned may not always transfer. Domain knowledge can, in part, inform

decisions, but as we showed the most intuitive representation may not be the best. It is important to test for invariances and evaluate the effectiveness of different representations. Going forward, we encourage researchers using other forms of spatiotemporal data to think carefully about the assumptions encoded by their model choices.

References

[1] T. Raiko, H. Valpola, and Y. LeCun, "Deep learning made easier by linear transformations in perceptrons," in *Artificial Intelligence and Statistics*, 2012, pp. 924–932.

[2] P. Sturmfels, S. Rutherford, M. Angstadt, M. Peterson, C. Sripada, and J. Wiens, "A domain guided cnn architecture for predicting age from structural brain images," *arXiv preprint arXiv:1808.04362*, 2018.

[3] G. Ning, Z. Zhang, and Z. He, "Knowledge-guided deep fractal neural networks for human pose estimation," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1246–1259, 2018.

[4] F. Ma, J. Gao, Q. Suo, Q. You, J. Zhou, and A. Zhang, "Risk prediction on electronic health records with prior medical knowledge," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1910–1919.

[5] J. Wang, I. Fox, J. Skaza, N. Linck, S. Singh, and J. Wiens, "The advantage of doubling: A deep reinforcement learning approach to studying the double team in the nba," *MIT Sloan Sports Analytics Conference*, 2018.

[6] M. Harmon, P. Lucey, and D. Klabjan, "Predicting shot making in basketball learnt from adversarial multiagent trajectories," *arXiv preprint arXiv:1609.04849*, 2016.

[7] D. Cervone, A. D'Amour, L. Bornn, and K. Goldsberry, "Pointwise: Predicting points and valuing decisions in real time with nba optical tracking data," in *MIT Sloan Sports Analytics Conference*, vol. 28, 2014, p. 3.

[8] D. Cervone, A. D'Amour, L. Bornn, and K. Goldsberry, "A multiresolution stochastic process model for predicting basketball possession outcomes," *Journal of the American Statistical Association*, vol. 111, no. 514, pp. 585–599, 2016.

[9] T. P. Lawler and F. H. Lawler, "Left-handedness in professional basketball: prevalence, performance, and survival," *Perceptual and motor skills*, vol. 113, no. 3, pp. 815–824, 2011.

[10] N. Mehrasa, Y. Zhong, F. Tung, L. Bornn, and G. Mori, "Deep learning of player trajectory representations for team activity analysis," in *MIT Sloan Sports Analytics Conference*, 2018.

[11] K.-C. Wang and R. Zemel, "Classifying nba offensive plays using neural networks," in *MIT Sloan Sports Analytics Conference*, 2016.

[12] G. Thomas, R. Gade, T. B. Moeslund, P. Carr, and A. Hilton, "Computer vision for sports: current applications and research topics," *Computer Vision and Image Understanding*, vol. 159, pp. 3–18, 2017.

[13] A. McIntyre, J. Brooks, J. Guttag, and J. Wiens, "Recognizing and analyzing ball screen defense in the nba," *MIT Sloan Sports Analytic Conference*, vol. 1001, p. 48104, 2016.

[14] H. M. Le, P. Carr, Y. Yue, and P. Lucey, "Data-driven ghosting using deep imitation learning," *MIT Sloan Sports Analytics Conference*, 2017.

[15] P. Lucey, A. Bialkowski, P. Carr, Y. Yue, and I. Matthews, "How to get an open shot: Analyzing team movement in basketball using tracking data," in *MIT Sloan Sports Analytics Conference*, 2014.

[16] A. Bocskocsky, J. Ezekowitz, and C. Stein, "The hot hand: A new approach to an old 'fallacy'," in *MIT Sloan Sports Analytics Conference*, 2014, pp. 1–10.

[17] L. J. Martin, P. Ammanabrolu, X. Wang, W. Hancock, S. Singh, B. Harrison, and M. O. Riedl, "Event representations for automated story generation with deep neural nets," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[18] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in neural information processing systems*, 2015, pp. 2224–2232.

[19] M. Kempe, A. Grunz, and D. Memmert, "Detecting tactical patterns in basketball: Comparison of merge self-organising maps and dynamic controlled neural networks," *European journal of sport science*, vol. 15, no. 4, pp. 249–255, 2015.

[20] D. Cervone, L. Bornn, and K. Goldsberry, "Nba court realty," in *MIT Sloan Sports Analytics Conference*, 2016.

[21] J. Wiens, G. Balakrishnan, J. Brooks, and J. Guttag, "To crash or not to crash: A quantitative look at the relationship between offensive rebounding and transition defense in the nba," in *MIT Sloan Sports Analytics Conference*, 2013.

[22] A. Miller, L. Bornn, R. Adams, and K. Goldsberry, "Factorized point process intensities: A spatial analysis of professional basketball," in *International Conference on Machine Learning*, 2014, pp. 235–243.

[23] A. Franks, A. Miller, L. Bornn, K. Goldsberry *et al.*, "Characterizing the spatial structure of defensive skill in professional basketball," *The Annals of Applied Statistics*, vol. 9, no. 1, pp. 94–121, 2015.

[24] A. A. Sangüesa, T. B. Moeslund, C. H. Bahnsen, and R. B. Iglesias, "Identifying basketball plays from sensor data; towards a low-cost automatic extraction of advanced statistics," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2017, pp. 894–901.

[25] Y. Yue, P. Lucey, P. Carr, A. Bialkowski, and I. Matthews, "Learning fine-grained spatial models for dynamic sports play prediction," in *2014 IEEE International Conference on Data Mining*. IEEE, 2014, pp. 670–679.

[26] J. D. Brooks, "Using machine learning to derive insights from sports location data," Ph.D. dissertation, Massachusetts Institute of Technology, 2018.

[27] Y.-W. Chao, S. Vijayanarasimhan, B. Seybold, D. A. Ross, J. Deng, and R. Sukthankar, "Rethinking the faster r-cnn architecture for temporal action localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1130–1139.

[28] Y. Zhao, R. Yang, G. Chevalier, R. C. Shah, and R. Romijnders, "Applying deep bidirectional lstm and mixture density network for basketball trajectory prediction," *Optik-International Journal for Light and Electron Optics*, vol. 158, pp. 266–272, 2018.

[29] R. Shah and R. Romijnders, "Applying deep learning to basketball trajectories," *arXiv preprint arXiv:1608.03793*, 2016.

[30] B. Normalization, "Accelerating deep network training by reducing internal covariate shift," *CoRR.–2015.–Vol. abs/1502.03167.–URL: http://arxiv. org/abs/1502.03167*, 2015.

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

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.

[33] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

Tab. 2: Player order does not appear to affect performance for time-series representation. AUC@0-2 denotes the average macro AUC evaluated at prediction horizon of 0-2 seconds. AUC@3-6 denotes the average macro AUC evaluated at prediction horizon of 3-6 seconds. We show median, (min, max) results across runs.

model	**AUC@0-2**	**AUC@3-6**
arbitrary	0.618 (0.605, 0.618)	0.504 (0.501, 0.506)
position	0.609 (0.608, 0.628)	0.501 (0.496, 0.509)
random permutation	0.617 (0.597, 0.620)	0.505 (0.495, 0.512)

Tab. 3: Player order does not appear to affect performance for flat representation. AUC@0-2 denotes the average macro AUC evaluated at prediction horizon of 0-2 seconds. AUC@3-6 denotes the average macro AUC evaluated at prediction horizon of 3-6 seconds. We show median, (min, max) results across runs.

model	AUC@0-2	AUC@3-6
arbitrary	0.628 (0.625,0.628)	0.510 (0.509, 0.531)
position	0.626 (0.624,0.632)	0.522 (0.519, 0.528)
random permutation	0.635 (0.624,0.638)	0.522 (0.515, 0.530)

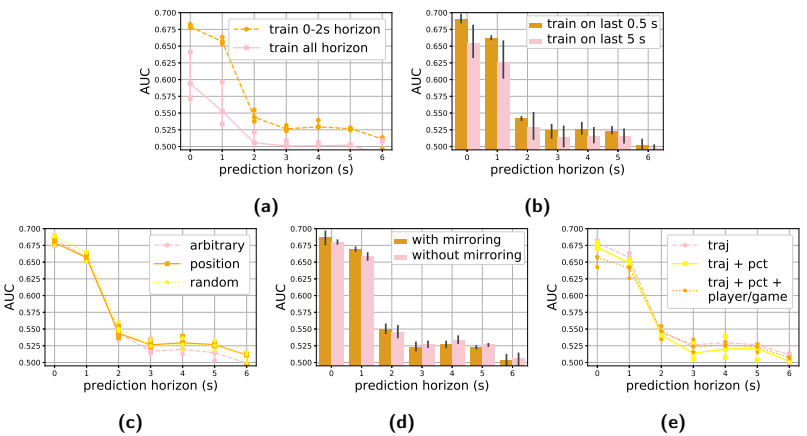

Fig. 6: Flat input results with macro-averaged AUC. (a) Training using a 0-2s prediction horizon leads to better generalization performance compared to using all prediction horizons. (b) Additional training data hurts model performance. (c) Player ordering does not affect performance. (d) Mirroring the court consistently helps training. (e) A flat input does not capitalize on the game information.

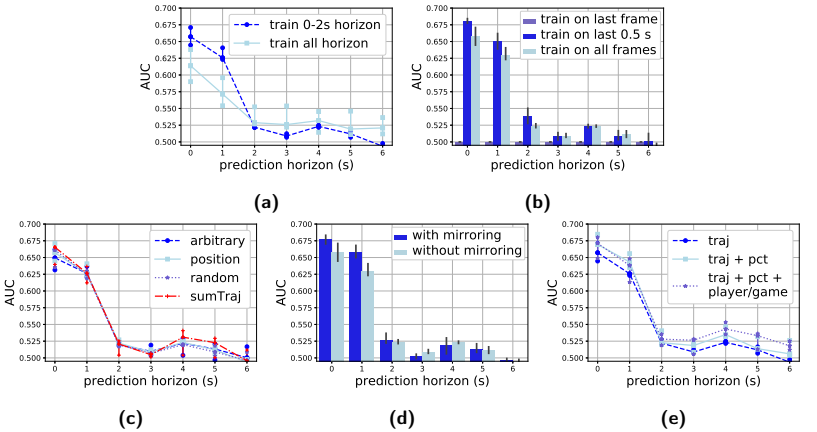

Fig. 7: Image input results with macro-averaged AUC. (a) Training using 0-2s prediction horizons generalizes better on shorter and worse on longer prediction horizons. (b) Long trajectories hurt performance, but 1 frame of information is not enough for effective image based representation. (c) Player ordering does not affect performance. (d) Mirroring the court helps training. (e) Game information helps for long prediction horizons.

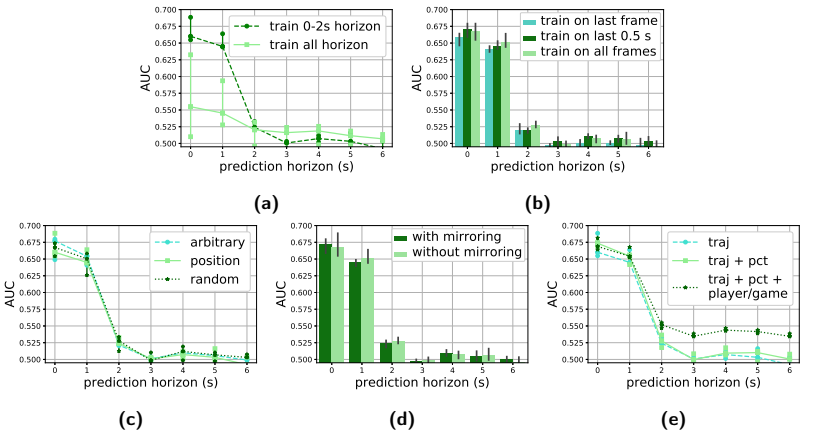

Fig. 8: Time-series input results with macro-averaged AUC. (a) Training using 0-2s prediction horizons generalizes better on short and worse on longer horizons. (b) Additional trajectory history has little effect on model performance. 1 frame of information is enough for effective representation (though performance decreases). (c) Player ordering does not affect performance. (d) Mirroring the court consistently helps training. (e) Game information consistently helps for larger prediction horizons.

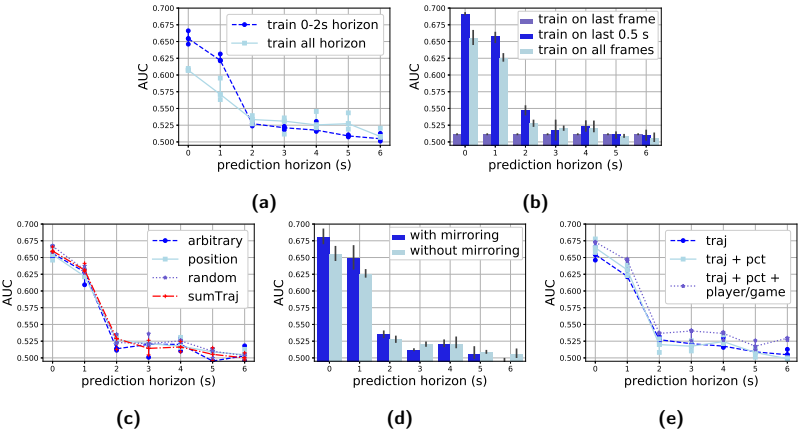

Fig. 9: Results using an image-based representation that crops the image to only the half-court with macro-averaged AUC. We perform the same experiments as in Figure 7. The results are similar to using full court image. We observe a slight increase in performance, likely due to a reduction in the sparsity of the input.