On Predicting and Generating a Good Break Shot in Billiards Sports

Qianru Zhang[*][†], Zheng Wang[*][‡], Cheng Long[‡], Siu-Ming Yiu[†]

Abstract

With the proliferation of tracking devices such as cameras and/or GPS sensors, sports data is being generated at an unprecedented speed and the interest in collecting some data from sports games has grown dramatically as well. The collected data facilitates various sports analytic tasks; however, these studies are mainly concerning with sports such as football and basketball. It remains largely unexplored for billiards sports though it is a popular sport of both strategy and physical skill, and this is mainly due to the lack of publicly available datasets. Motivated by this, we collect a dataset of billiards sports, which includes the layouts (i.e., locations) of billiards balls after performing break shots, called break shot layouts, the traces of the balls as a result of strikes (in the form of trajectories), and detailed statistics and performance indicators. On top of the dataset, we investigate several tasks, including prediction and generation on the layouts data and similarity search on the trajectory data, which can serve different users such as coaches, players and fans. We conduct extensive experiments on the collected dataset for the tasks, and the results demonstrate the superior performance of the methods proposed in this paper.

1 INTRODUCTION

Nowadays, it becomes a common practice to collect some data from sports games using tracking devices such as cameras and/or GPS sensors. The collected data facilitates various analytic tasks such as similar game retrieval [26, 22], tactics detection [8] and score prediction [2, 30], which cover sports including basketball, soccer, volleyball and handball. Yet these analytic tasks have not been explored on billiards sports, and this is mainly due to the lack of publicly available datasets of billiards sports. The billiards sport is a two-player game. In each game, there are a certain number of rounds (called *frames*). In each frame, two players take turns to strike a billiards ball (called the cue ball) so as to pot other balls (called object balls) to the pockets of a pool table (See Figure 1 for example). The object balls are usually associated with numbers and/or colors. There are multiple types of billiards sports (e.g., 9-ball and Snooker), and for different types, there are different rules of striking object balls and also criteria of winning a frame. For example, for 9-ball, a player needs to hit the object ball with the smallest number among those remaining on the table for each strike and wins the frame if he/she pots the object ball with the number 9. Billiards sport, which can be played in halls, hotels, or as often in the club houses of other organisations, becomes a popular recreational sport [25, 19].

Motivated by this, we propose to collect some data capturing billiards games so that various types of data analyses can be conducted on the data for discovering knowledge about the games and players. Similar to other sports such as football and basketball, billiards sports are usually recorded as videos, which are then made publicly available on the Web. A natural idea is to extract data of billiards sports from videos. In particular, we collected the dataset from 9-ball games (i.e., one popular type of billiards sports), which are governed by the World Pool-Billiard Association (WPA) since 1990 for around 30 years up to now. The dataset covers games of 94 international professional 9-ball tournaments in the most recent two decades, which were played by 227 professional players. In summary, the dataset includes 3,019 records for frames, 6,637 records for turns, and 2,082 records for strikes. More detailed summarization of the dataset could be found in Table 1 and description in the supplementary materials. To our best knowledge, this is the first billiards dataset that will be made publicly available.

On top of the dataset, there are many potential analytic tasks that could be explored such as similarity search, tactics discovery, game result prediction, player performance analysis, etc. We have explored three tasks including prediction, generation and similarity search on the collected dataset, with the former two on the break shot layout data and the last on the trajectory data of strikes. Due to the page limit, we present the first two in this paper and the last in the supplementary materials.

Break Shot Layout Prediction. The first task is a *prediction* task. Intuitively, the layout of a break shot embeds rich information of the game, and based

[*] Equal contribution.
[†] The University of Hong Kong, {qrzhang, smyiu}@cs.hku.hk
[‡] Nanyang Technological University, {wang_zheng, c.long}@ntu.edu.sg

on the layout, we predict three aspects of the game: (1) clear or not, it predicts whether the player who performs the break shot will pocket all balls; (2) win or not, it predicts whether the player who performs the break shot will win the game; (3) it predicts how many balls are consecutively potted after the break shot. The prediction task is a fundamental functionality with many application scenarios. One scenario is to provide a real-time prediction of the winning result of a game based on the layout. This prediction functionality is highly desirable for broadcasting platforms of the billiards games such as ChalkySticks [1]. Another scenario of the prediction functionality is to serve a billiards coach to analyze how well a player performs and provide instant feedback on the break performance for players. The prediction task is non-trivial, which is mainly due to the characteristics of the data. In particular, the billiards layout data is unlike other existing datasets including a set of points [31, 23] or a sequence of locations such as trajectory data [33]. In a billiards layout, the billiards balls including one cue (white) ball and several object (colored) balls are all located on a rectangular pool table with six pockets. The data embeds rich correlations, e.g., the spatial correlations between the cue ball and object balls, and the correlations between the cue ball (or object balls) and the six pockets, which is unique and important in billiards sports. In this paper, we carefully extract those unique features that can reflect the spatial correlation of balls and/or pockets. Then, we use Convolutional Neural Network (CNN) to accomplish this task in a supervised learning manner, since CNN is inherently applicable for perceiving the correlations.

Break Shot Layout Generation. The second task is a *generation* task, which is to generate realistic yet high-quality (i.e., easy-to-clear) layouts of break shots. For a player, it is critical to gain knowledge about such layouts that (1) he/she can accomplish them (i.e., the layouts are realistic) and (2) he/she would pot many balls and even clear the table given them (i.e., the layouts are of high quality). This task would be very useful for the player to enrich his knowledge by generating more layouts of such kind. To achieve the task, we explore a data-driven solution to generate the billiards layouts via Generative Adversarial Networks (GANs) [12], called BLGAN. More specifically, we treat each billiards layout as an ordered sequence, i.e., one cue ball followed by several object balls that remain on the table in ascending order of their numbers. Then, a generator is used to generate a sequence of discrete tokens, each representing a discretized location of the billiards table (e.g., a grid cell). In addition, to generate *realistic* layouts that can be accomplished by a player, we collect the real layouts with the clear labels from the dataset to guide the training of the generator. In this way, the layouts are generated with some real layouts but not random ones inputted as seeds, and thus we believe the generated layouts tend to be realistic ones. For the discriminator, we also collect some real layouts that are predicted to be cleared with high probabilities. These real layouts and the generated layouts together are fed to the discriminator, which tries to discriminate between the real ones and generated ones. The discriminator provides feedback to encourage the learning of the generator, i.e., to generate layouts that are difficult to be discriminated from those real layouts that are predicted to be cleared with high probabilities. In this way, the generated layouts tend to be high-quality ones.

Contributions. The main contributions of this paper are summarized as follows. (1) We contribute the first billiards sports dataset, which includes break shot data (for frames), strike statistics data (for turns) and trajectory data (for strikes). The layouts data is a new data type, which is related to yet different from quite a few existing data types including point sets, trajectories, sequences, etc. Specifically, it consists of a point set, is order-sensitive, and is associated with some contexts (e.g., the pockets). To the best of the authors' knowledge, this would be the first dataset of its kind. In addition, the dataset has rich contents and labels information to support various machine learning tasks. Our dataset and codes are publicly accessible via the link [2] and provide an opportunity for research communities such as machine learning, data mining, computational geometry, computer vision and sports science, to make a significant impact. (2) We investigate three important tasks on the top of the dataset, including prediction and generation on break shot layouts data and similarity search on trajectory data of strikes. These tasks help better understand the sports and serve different users including coaches, players and fans. (3) Extensive empirical evaluations are performed on the collected dataset for the three tasks, demonstrating superior performance over baseline methods.

2 RELATED WORK

Sports Data Analytics. Billiards corresponds to one type of popular sports playing with a cue stick on a pool table. The traditional research in this area addresses the task of training the robotic players such as Deep Green [18] to execute good shots on a physical table and thus win a computer billiards game [24, 3, 4]. Nowadays,

[1]https://www.chalkysticks.com/tv

[2]https://drive.google.com/drive/folders/1NBqonYLr_cParMMn4xSeE0KTJNhjeYuG?usp=sharing

it becomes a common practice to deploy some devices such as GPS sensors and cameras to capture some data of sports games, and the collected data proliferates various sports analytic tasks. Specifically, Wang et al. [26] study the similar soccer game retrieval, which can recommend similar games to sports fans in some sports recommender systems such as ESPN. Decroos et al. [8] collect event-stream data from professional soccer matches, and explore the problem of tactics detection, which helps players improve their tactics when they prepare for an upcoming match. Aoki et al. [2] collect four different sports data including basketball, soccer, volleyball and handball, and analyze the difficulty of predicting the outcome of sports events. Overall, these tasks involve various sports such as basketball, soccer, volleyball and handball.

Sequence Data Prediction. The collected billiards data contains the layouts of break shots in real-world 9-ball games, and it can be treated as data of sequences of one cue ball followed by several object balls in the ascending order of their numbers. We review the existing works of sequence data prediction as follows. The common sequence data includes time series data and trajectory data. Time series prediction [27, 11] is a related but different task. It aims to train models to fit historical data and use them to predict future observations of a sequence. Existing methods for time series prediction can be grouped into two categories: classical models such as SVM [21] and recent deep learning-based models such as RNN variants [16] or Transformer variants [34]. On the other hand, trajectory data corresponds to a sequence of positions to capture the traces of moving objects. The problem of trajectory prediction can be viewed as a sequence generation task of predicting the future trajectories of moving objects based on their past positions, and it is widely used to avoid obstacles for pedestrians [1, 14] and autonomous vehicles [9]. Our task differs from these studies mainly in two aspects. First, our billiards data carries the unique characteristics in billiards sports, e.g., it captures the locations of balls on a pool table. Second, our task is to predict some associated information with the layout instead of forecasting a future observation value in a sequence.

Sequence Generative Models. We review existing generative models [29, 7, 10, 17, 20] related to the task of break shot layout generation as follows. The majority of deep generative models are used to generate text. For example, SeqGAN [29] is a typical adversarial text generation model, which builds an unbiased estimator based on the REINFORCE algorithm [28] for the generator, and applies the roll-out policy to obtain the reward from the discriminator. In addition, to deal with the differentiation difficulty when applying GAN to generate sequences of discrete elements, GSGAN [15] is proposed to use Gumbel-softmax output distributions to train GAN on discrete sequences. Guo et al. [13] propose the LeakGAN, which introduces a hierarchical reinforcement learning framework for the generator, and improves the performance of long text generation. Overall, all of these works consider textual data, e.g., generating some sentences. Our work differs from them mainly in the data, i.e., billiards layout data corresponds to a set of billiards balls that are located on a pool table. Besides, the data is associated with several unique characteristics in billiards sports, e.g., the spatial correlations between the cue (white) ball and object (colored) balls.

3 DATASETS

Our real-world billiards data is extracted from the videos of all professional billiards games published on YouTube in the most recent two decades using the software Kinovea (https://www.kinovea.org/). The data collection process by the software Kinovea consists of four steps (i.e., uploading images/videos, adding grids and markers, exporting coordinates and collecting information), which are illustrated in details in the supplementary materials. The dataset covers 227 players and 94 international professional 9-ball tournaments. We collect the billiard dataset for frames, turns and strikes. Table 1 summarizes the collected dataset (with details included in the supplementary materials).

4 TASKS

4.1 Layout Prediction with BLCNN In this section, we introduce how to embed each ball in billiards layouts into real vectors. Then, the vectors are concatenated in the order of balls that should be potted as an embedding of features, which is fed into a classifier based on the architecture of CNN to predict the results. The effectiveness of prediction depends on the quality of extracted features. The proposed model for billiards layout prediction is called BLCNN.

Feature Extraction. We capture the correlations in a billiard layout and consider the following three kinds of features: the location information of billiards balls on the table, called Ball-Self (BS); the correlation between balls and pockets, called Ball-Pocket (BP); the correlation between balls, called Ball-Ball (BB). Figure 1 illustrates these features from a 9-ball billiards layout.

(1) Ball-Self (BS). We map the play field of the billiards table into a 200× 100 coordinate system and set the bottom left corner as the origin. Thus, the range of the x-axis of the coordinate system is [0,200], the range of

Table 1: Summary of the collected billiards dataset (cover 94 tournaments and 227 palyers).

Data of Frames (3,019)		Data of Turns (6,637)		Data of Strikes (2,082)			
1	break shot layout	1	player name	1	trajectory	6	stick top position
2	three performance indicators: clear label, win label and # of potted balls label	2	# of strikes	2	camera angle		
		3	order of potted balls	3	cushion	7	direction on hitting
		4	type of foul	4	intersection point		
		5	data of strikes	5	cue ball position		

the y-axis of the coordinate system is [0,100]. For each ball b_i in the billiards layout, it is related to (x, y) in the coordinate system, which is regarded as its position feature on the pool table. For example, in Figure 1, the location of ball 2 (blue) is $(88.06, 76.02)$.

(2) Ball-Pocket (BP). To report features between each ball and each pocket, we set the pocket at the bottom left corner as the start pocket and mark the pockets clockwise from 1 to 6, which is shown in Figure 1. For each ball b_i, we use four features including cushion angle, distance to pocket, an indicator of occlusion and pocket index to describe it wrt each pocket $p_j (1 \leq j \leq 6)$ on the table. Thus, there are in total $4 \times 6 = 24$ features for each ball b_i. (i) Cushion angle. Cushion angle is used to describe the minimum angle between the line from a ball b_i to a pocket p_j (i.e., denoted by $\overline{b_i p_j}$) and its two cushion edges. (ii) Distance to pocket. Distance to pocket is the distance from a ball b_i to a pocket p_j (i.e., $\|b_i - p_j\|_2$). (iii) Indicator of occlusion. An indicator of occlusion is used to report whether there is a ball on the line from a ball b_i to a pocket p_j. If so, the indicator of occlusion is 1; 0 otherwise. (iv) Pocket index. The pocket index is used to distinguish different called pockets, which is from 1 to 6. Take ball 2 as an example in Figure 1, the cushion angle is 40.80° between the line $\overline{b_2 p_1}$ and cushion 1-6; the distance from ball 2 to pocket 1 is 116.33; the indicator of occlusion is 1 since cue ball appears in the path from ball 2 to pocket 1; the pocket index is 1 for the above three features. In brief, the feature (i) and (ii) capture the feasibility to sink the ball 2 into a called pocket; the feature (iii) captures whether there will be collisions or bounces on the path of ball 2 to a called pocket; the feature (iv) is to distinguish different called pockets varying from 1 to 6.

(3) Ball-Ball (BB). We consider two features to capture the correlation of balls, including shot angle and pocket index. (i) shot angle. Shot angle describes the minimum angle between the line of two balls with consecutive numbers on the table (i.e., denoted by $\overline{b_i b_{i+1}}$) and the line from ball b_{i+1} to each pocket p_j. In Figure 1, the shot angle is 8.11° since the minimum angle corresponds to the angle of a line from the cue ball to ball 2 (i.e., $\overline{b_1 b_2}$) and the line from ball 2 to pocket 3. (ii) Pocket

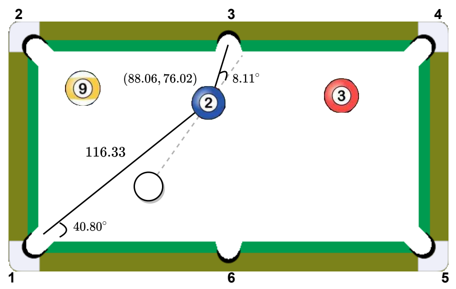

Figure 1: Illustration of the BS, BP and BB features of ball 2 (blue) in a layout.

index. The pocket index is selected to distinguish the index of the most likely called pocket when performing a strike. In the case of ball 2 in Figure 1, the pocket index is 3 since pocket 3 is more likely to be called with the minimum shot angle 8.11°.

CNN-based Classifier. BLCNN for the prediction task is a CNN-based architecture, which is illustrated in Figure 2. We consider three labels in the task including clear or not, win or nor and the number of potted balls after the break shot based on a given billiards layout.

In particular, for each billiards ball, we transform its extracted features (i.e., BS, BB and BP) into real vectors via an embedding layer, then the vectors are concatenated into a long vector as the embedding of the ball. To achieve the embedding, we granulate these features into the tokens, of which the details are provided in Section 5.1. Next, the ball embeddings are further concatenated in the ascending order of their numbers and we apply padding when there are missing balls on the table. Thus, we get an embedding for the features of billiards balls in a matrix form, called embedded features. Note that we embed these features via an embedding layer instead of using the feature values directly because the feature values restrict the embeddings in a low dimensional space, which makes it difficult for the model to be further optimized. Further, the embedded features are fed to a convolutional layer with multiple filters of varying sizes to obtain feature maps, and a global max-pooling layer is then employed to capture the most important feature for each feature map, and those important features are further aggregated into

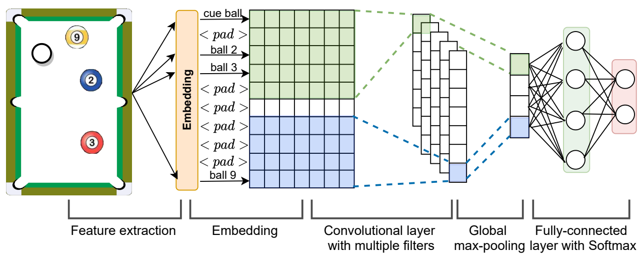

Figure 2: The model architecture of BLCNN, where the colours (i.e., green and blue) denote the convolutional filters with different sizes to obtain feature maps.

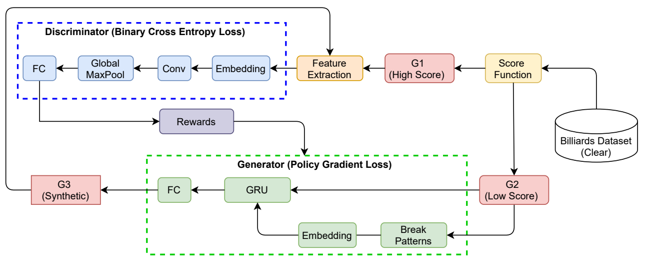

Figure 3: The model architecture of BLGAN. The score function is to identify two groups of layouts with high and low scores (i.e., G_1 and G_2) from the dataset. In discriminator, it follows the architecture in BLCNN, and provides the rewards outputted via a fully-connected (FC) layer for training the generator. In the generator, the break patterns are extracted from G_2, each pattern corresponds to a token, which is fed into an embedding layer to obtain a latent vector as the initial hidden vector, to generate the G_3 via GRU followed by a FC layer.

a vector, which corresponds to a representation of the billiards layout. The layout representation is then fed to a standard neural network architecture, i.e., the fully-connected layer with softmax nonlinearity to produce the output. We train the BLCNN in a supervised learning manner, with the cross-entropy loss for the prediction of three types of labels (i.e., clear, win and potted balls).

4.2 Layout Generation with BLGAN We first explain that the layout generation task is useful by referring to some existing literature of billiards sports [24, 3, 4], which aim to find high-quality break shots so as to achieve desirable layouts. For example, in [24], it models a layout (i.e., the positions of balls in a billiards table) as a state and how to execute a shot as an action. Each action is controlled by five continuous parameters including the direction of hitting, etc. It aims to search for a shot that would generate a desirable layout. It is also worth mentioning that our dataset includes sufficient information on the the strike level (including the break shots), such as the hitting position, stick top position, direction on hitting, which we believe will facilitate more in-depth research along this line.

Challenges. Generative Adversarial Network (GAN) [12] is a promising framework to generate new data based on existing data. A natural idea is to treat a billiards layout as an ordered sequence, i.e., one cue ball followed by object balls that remain on the table in ascending order of their numbers, and apply GAN to generate sequences of balls (and their locations). Nevertheless, it has the following challenges. First, when the physical space of the billiards balls (i.e., the billiards table) is discretized (e.g., as a 15×15 grid), applying GAN to generate the balls and their locations (as discrete tokens) would suffer from a classical issue of indifferentiability [15], since the samples from a distribution of discrete tokens are not differentiable with respect to the distribution parameters. Second, in a break shot layout, some balls may have been potted already, and in this case, the player who performs the break shot can keep his turn and thus have the chance to clear the table given the layout. Yet a straightforward GAN does not provide a mechanism of deciding some ball/balls to be omitted from the generated layouts, i,e., meaning some of the balls are potted already.

Overview. To serve the target of generating the layouts that are more likely to be cleared, we collect all layouts with clear labels from the dataset. We employ a *score function* (more details would be discussed later) to score the quality of a layout, sort the layouts in descending order of their scores and divide the billiards layouts equally into two groups without overlapping (i.e., one group (G_1) represents layouts with high scores (high-score), the other group (G_2) represents layouts with low scores (low-score)). G_2 is fed to the *generator* in BLGAN to guide the new layout generation denoted by G_3. Also, we train a *discriminator* to provide guidance for improving the quality of billiards layouts the generator generates by feeding the examples of G_1 (i.e., the real layouts with high scores) and the examples of G_3 (i.e., the generated layouts). The GAN-based architecture is illustrated in Figure 3. We introduce the detailed designs next.

Score Function. We use the BLCNN model to compute the score of a layout since BLCNN model can predict whether a layout will be cleared or not, which indicates the quality of the layout. BLCNN involves a fully-connected layer with the softmax function and thus it computes scores from 0 to 1.

Generator. To overcome the issue of indifferentiability, we follow an existing strategy [29, 5, 6] to model the sequence generation procedure as a sequential decision making process, which is optimized by the REINFORCE algorithm via Policy Gradient [28] and thus it naturally bypasses the differentiation difficulty for discrete tokens. In particular, we train a generative model G to generate a sequence $C_{1:T} = (c_1, c_2, ...c_t, ...c_T)$, $c_t \in \gamma$, where c_t denotes the location token of a ball, and γ corresponds to the vocabulary of all location tokens on the billiards table. At time step t, the state s corresponds to the sequence of generated tokens so far, i.e., $(c_1, c_2, ...c_{t-1})$; and the action a is to select the next location token (i.e., c_t) based on the state s. To obtain the reward r, the existing study [29] adopts the roll-out policy by sampling the unknown tokens in a sequence to obtain an immediate reward from the discriminator. In this paper, we adopt a simpler strategy, i.e., we collect a reward only after an entire layout is generated and fed into the discriminator - recall the discriminator would return a score which is used as the reward signal, and it is shared to all steps when the entire layout has not been generated. In addition, we collected a set of patterns, each corresponding to a set of balls of a layout, which appear in the real dataset. Each such pattern is called a break pattern and is encoded using one-hot vectors. For example, consider a break shot layout in a 9-ball game with the cue ball and ball 1, 2, 3, 5, 7, 8, 9 on the pool table. The break pattern of this layout is denoted by $(4, 6)$ since ball 4 and 6 are potted already and thus missing from the layout. When generating a layout, we randomly sample a known break pattern and feed its vector to the generator as an initial hidden vector to guide the generation task. This is to handle the second challenge.

Discriminator. By empirical findings, the CNN-based model has shown good performance to capture the correlations for the billiards layout data, e.g., it achieves the accuracy around 90% on the previous prediction tasks. Thus, we adopt the CNN-based architecture in the discriminator, whose main function is to discriminate the generated layouts (i.e., G_3) and high-score real layouts (i.e., G_1), and provide a reward signal to guide the learning of the generator, i.e. it encourages the generator to generate the layouts of high-score (i.e., G_3) from those low-score real layouts (i.e., G_2). The optimization of the discriminator is to minimize Binary Cross-Entropy (BCE) between G_1 and G_3.

5 EXPERIMENTS

5.1 Evaluation on the Prediction Task To embed each feature, we partition the play field of the billiards table into grids with a predefined granularity 15×15, and thus we got 98 unique location tokens for BS features given that the coordinate system on the pool table is 200×100. We will study the effect of different grid cell sizes later on. For BP and BB features, we partition the angle space and the distance space with the resolutions of $15°$and 10, respectively. We tried different partitions since the results are similar and thus omitted. For each feature, we embed it into a 10-dimensional vector, and thus each ball corresponds to a 270-dimensional vector, i.e., $10 * (1 + 4 * 6 + 2) = 270$. In particular, for each ball, there is one token which corresponds to a cell on the pool table for the BS feature; four tokens which correspond to cushion angle, distance to pocket, indicator of occlusion and pocket index for the BP feature for each of the six pockets; and two tokens which correspond to shot angle and pocket index for the BB feature. To obtain the representation of each layout, we employ a convolutional layer with varying filter sizes from (1, 270) to (10, 270). Note that larger filters are helpful for capturing the correlations between the balls. Then a global max-pooling layer is used to generate a 30-dimensional vector as the representation of each layout. We randomly sample 40% layouts from the dataset for training, which is enough for achieving a satisfactory result based on empirical findings, and the remaining is for evaluation.

Baselines. Based on the extracted BS, BP and BB features, we consider 11 classifiers including BLCNN, LSTM, RBFSVM, LinearSVM, Logistic Regression, Decision Tree, Random Forest, Bayes, Adaboost, Gradient Boost and XGBoost on three prediction tasks. We also compare a heterogeneous graph neural network-based method called HetGNN [32]. Specifically, we consider two types of nodes (i.e., pockets and balls) in a billiards layout, which interact with one another. The representation of the whole layout is calculated as the average of all node embeddings, and we follow the original paper by using the Logistic Regression classifier for the prediction.

Prediction evaluation. We report the accuracy on three prediction tasks in Figure 4. We observe BLCNN has the best performance with the accuracy 89.69%, 86.56% and 80.94% on the three tasks. We also notice that the accuracy on predicting the number of potted balls is inferior to the prediction of clear or win, since predicting the number of potted balls is a multi-class classification problem; while predicting clear or win is a classical binary classification problem. Besides, in the prediction task of potted balls, we observe the accuracy of the Decision Tree is 4.21%, which is caused by unbalanced labels. For example, there are only 30 billiards layouts with label 4 while there are 905 billiards layouts with label 0. We also notice the performance of HetGNN is consistently inferior to BLCNN on three tasks, which indicates the features we extracted are useful for the prediction tasks.

Parameter study. To evaluate the effect of cell size on prediction tasks, we conduct our experiments by

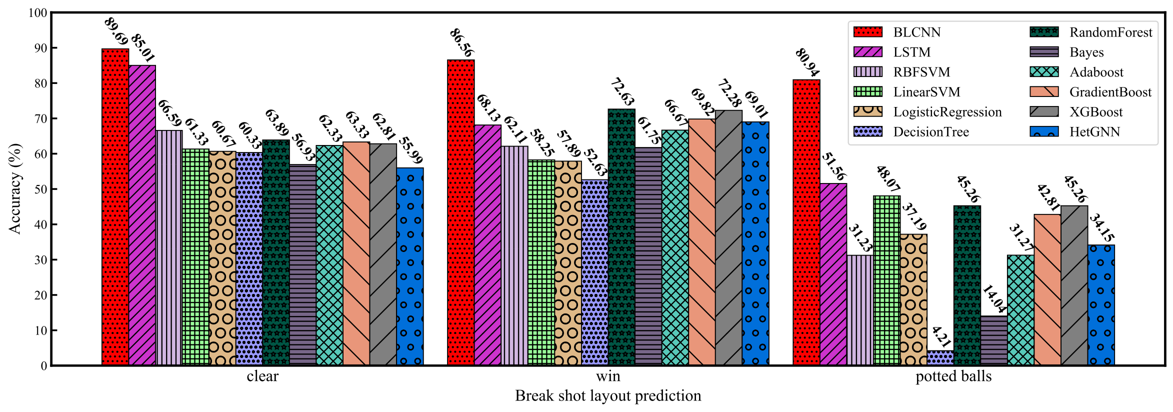

Figure 4: Prediction results.

Table 2: Effect of the cell size of BLCNN.

Cell	#Tokens	Prediction(%)		
		Clear	Win	Potted balls
10	200	84.06	84.69	76.88
15	98	**89.69**	**86.56**	**80.94**
20	50	84.69	84.06	73.75
25	32	83.44	80.00	79.69

Table 3: Ablation study of BLCNN.

Model	Prediction(%)		
	Clear	Win	Potted balls
BLCNN	**89.69**	**86.56**	**80.94**
w/o BS	88.44	85.50	73.44
w/o BP	70.00	74.37	42.19
w/o BB	86.56	84.38	76.25

varying the sizes from 10 to 25. In Table 2, we report the performance of our BLCNN on the three prediction tasks. When the cell size is fixed, the accuracy of potted balls is worse than the accuracy of clear or win because the model for the former problem is more difficult to train. We set the cell size to 15 because it provides the best performance in general.

Ablation study. To evaluate the importance of each component of extracted features in BLCNN, including Ball-Self (BS), Ball-Pocket (BP) and Ball-Ball (BB). We perform an ablation experiment and denote BLCNN without BS, BP and BB as w/o BS, w/o BP and w/o BB, respectively. Table 3 shows different versions for three prediction tasks. Overall, all these components are helpful to enable the model to achieve superior performance than other classifiers. The details on the effectiveness of each component are discussed as follows. (1) BS captures the spatial information of each ball. Thus when it is omitted, prediction performance on potted balls drops significantly by around 9.3%. (2) BP captures correlations between the ball to each pocket. It consists of most of the tokens (i.e. $4 \times 6 = 24$ features) in a total of 27 tokens to represent each ball, and therefore the BP features contribute the most for all of the tasks. As expected, if the BP features are omitted, the performance of prediction tasks drops by 21.9% (resp. 14.1% and 47.9%) when predicting clear (resp. win and potted balls). (3) BB captures the correlation between balls in the layout. It also affects the overall performance. For example, when BB features are omitted, the performance of BLCNN on potted balls drops by 5.7%.

5.2 Evaluation on the Generation Task Our method BLGAN consists of a generator and a discriminator. We use GRU as the generator to generate a sequence of locations of balls. We use the architecture of BLCNN as the discriminator since BLCNN has shown good performance on recognizing the layouts that are associated with the clear label with an accuracy of around 90%. The goal of BLGAN is to generate more layouts that are associated with the clear label. We randomly sample around 1200 cleared layouts to train BLGAN. Specifically, we divide the clear layouts into two groups, the first group has 600 layouts with higher scores (denoted by G_1) and the second group has another 600 layouts with lower scores (denoted by G_2). The convolutional layer in the discriminator is with the ReLU activation and 10 convolutional filters with varying sizes from (1, 270) to (10, 270), where 270 is the embedding dimensions of each ball as mentioned above. The fully connected layer involves one neuron with the sigmoid function to output the reward value.

Evaluation on quality and reality. Table 4 shows the average scores in terms of the real layouts in G_2 and the synthetic layouts in G_3, which involve the four most frequently occurring break patterns, meaning the patterns that occur more frequently compared to others in the dataset. We observe BLGAN has good

Table 4: Evaluation on quality (as scores) and reality (as relative ranks (distance)) for the four most frequent break patterns.

Break patterns	Pattern 1	Pattern 2	Pattern 3	Pattern 4
	(Ball 4 potted)	(Ball 5 potted)	(Ball 6 potted)	(Ball 1,5 potted)
G2 (real)	0.794	0.692	0.745	0.758
G3 (synthetic)	0.898	0.955	0.946	0.986
Relative rank (distance)	0.469	0.384	0.459	0.528

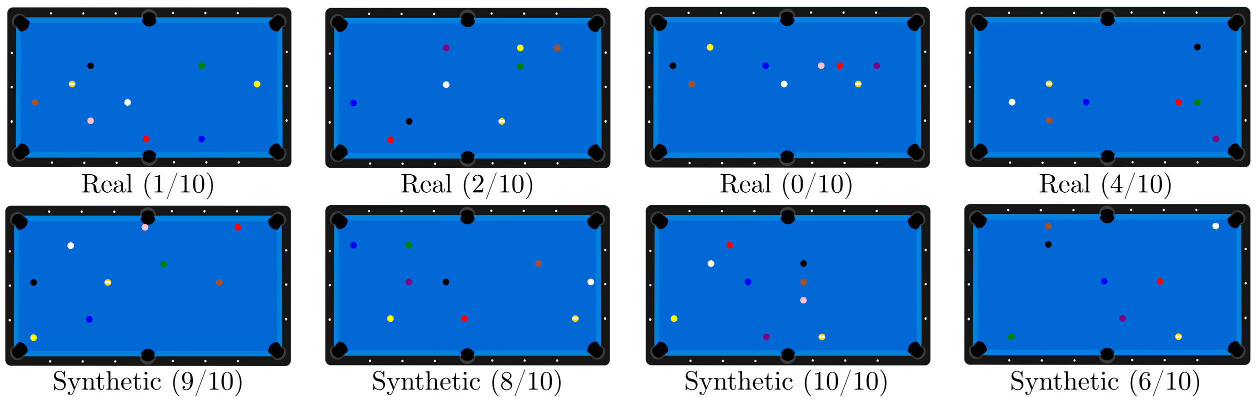

Figure 5: Top-4 break patterns (from left to right) for user study, where (1/10) means the 1 user votes this layout can be easily cleared among the total of 10 users.

performance on generating the layouts (i.e., G_3) of high quality, e.g., their scores are all-around 0.9 for the four break patterns in Table 4. We also investigate how similar the generated layouts are to the real ones. Specifically, for two layouts under the same break pattern, we measure the distance (dissimilarity) between them as the sum of the distances between their balls matched based on the numbers (e.g., the ball with number 1 in a layout is matched with the ball with number 1 in the other). Note that all balls can be matched for layouts under the same break pattern. Then, for a generated layout, we compute the average distance from its 5 nearest layouts among the real layouts under the same break pattern. Similarly, we compute the average distance for each real layout (from all other real layouts) under the same break pattern. We measure the relative rank of the average distance of the generated layout among all average distances, and the results are shown in Table 4. We observe that the generated layouts are in general quite similar to real ones, e.g., with the relative rank below 0.5 in most cases, where the relative rank (i.e., a normalized version of mean rank) below 0.5 means the layout we generated exceeds the similarity of more than half of the real layouts.

User study. In Figure 5, we conduct a user study to show the quality of the generated layouts. We randomly select a real layout from G_2 and a synthetic layout from G_3 that involve the four most frequent break patterns. We invite 10 volunteers with strong billiards knowledge to annotate which layout is more likely to be cleared. We first spend some time introducing some background to the volunteers so that they can understand these images in Figure 5. Note that the volunteers do not know which are real or synthetic. We observe BLGAN generates the layouts that can be easily cleared, i.e., the synthetic layouts get 82.5% votes while real layouts get only 17.5% votes from users. Take break pattern 1 (Ball 4 missed) as an example, we observe the synthetic layout can be easily cleared since Ball 1 (yellow) can be easily potted to the pocket at the bottom left corner at the beginning, and Ball 2 (blue) is located near the path of the cue ball.

6 CONCLUSION

In this paper, we contribute the first public available billiards layout dataset, which includes break shot layout data, trajectory data of strikes, details of strikes, etc. On the dataset, we investigate a few tasks including prediction and generation based on the break shot layout data and the similarity search based on the trajectory data. Extensive experiments are conducted on the collected dataset, which verify the usefulness of the dataset and also the proposed methods for the tasks on the dataset. We anticipate that our paper and dataset will boost more data analysis on billiards sport, including but not limited to player performance analysis, tactics discovery, similar layout retrieval, etc.

References

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *CVPR*, pages 961–971, 2016.

[2] R. Aoki, R. M. Assuncao, and P. O. Vaz de Melo. Luck is hard to beat: The difficulty of sports prediction. In *SIGKDD*, pages 1367–1376. ACM, 2017.

[3] C. Archibald, A. Altman, and Y. Shoham. Analysis of a winning computational billiards player. In *IJCAI*, volume 9, pages 1377–1382, 2009.

[4] C. Archibald and Y. Shoham. Modeling billiards games. In *AAMAS*, pages 193–199. Citeseer, 2009.

[5] P. Bachman and D. Precup. Data generation as sequential decision making. *arXiv preprint arXiv:1506.03504*, 2015.

[6] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, 2016.

[7] T. Che, Y. Li, R. Zhang, R. D. Hjelm, W. Li, Y. Song, and Y. Bengio. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*, 2017.

[8] T. Decroos, J. Van Haaren, and J. Davis. Automatic discovery of tactics in spatio-temporal soccer match data. In *SIGKDD*, pages 223–232. ACM, 2018.

[9] N. Deo and M. M. Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *CVPRW*, pages 1468–1476, 2018.

[10] W. Fedus, I. Goodfellow, and A. M. Dai. Maskgan: better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*, 2018.

[11] R. J. Frank, N. Davey, and S. P. Hunt. Time series prediction and neural networks. *JIRS*, 31(1):91–103, 2001.

[12] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[13] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang. Long text generation via adversarial training with leaked information. In *AAAI*, volume 32, 2018.

[14] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *CVPR*, pages 2255–2264, 2018.

[15] M. J. Kusner and J. M. Hernández-Lobato. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint*, 2016.

[16] Y. Li, R. Yu, C. Shahabi, and Y. Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *ICLR*, 2018.

[17] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun. Adversarial ranking for language generation. *NIPS*, 2017.

[18] Z. Lin, J.-S. Yang, and C. Yang. Grey decision-making for a billiard robot. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, volume 6, pages 5350–5355. IEEE, 2004.

[19] S. Mathavan, M. Jackson, and R. M. Parkin. A theoretical analysis of billiard ball dynamics under cushion impacts. *JMES*, 224(9):1863–1873, 2010.

[20] W. Nie, N. Narodytska, and A. Patel. Relgan: Relational generative adversarial networks for text generation. In *ICLR*, 2018.

[21] N. I. Sapankevych and R. Sankar. Time series prediction using support vector machines: a survey. *IEEE CIM*, 4(2):24–38, 2009.

[22] L. Sha, P. Lucey, Y. Yue, P. Carr, C. Rohlf, and I. Matthews. Chalkboarding: A new spatiotemporal query paradigm for sports play retrieval. In *IUI*, pages 336–347. ACM, 2016.

[23] K. Skianis, G. Nikolentzos, S. Limnios, and M. Vazirgiannis. Rep the set: Neural networks for learning set representations. In *AISTAT*, 2020.

[24] M. Smith. Running the table: An ai for computer billiards. In *Proceedings of the national conference on artificial intelligence*, volume 21, page 994. AAAI, 2006.

[25] M. Smith. Pickpocket: A computer billiards shark. *Artificial Intelligence*, 171(16-17):1069–1091, 2007.

[26] Z. Wang, C. Long, G. Cong, and C. Ju. Effective and efficient sports play retrieval with deep representation learning. In *SIGKDD*, pages 499–509, 2019.

[27] A. S. Weigend. *Time series prediction: forecasting the future and understanding the past.* Routledge, 2018.

[28] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[29] L. Yu, W. Zhang, J. Wang, and Y. Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, volume 31, 2017.

[30] Y. Yue, P. Lucey, P. Carr, A. Bialkowski, and I. Matthews. Learning fine-grained spatial models for dynamic sports play prediction. In *ICDM*, pages 670–679. IEEE, 2014.

[31] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. Deep sets. *arXiv preprint arXiv:1703.06114*, 2017.

[32] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla. Heterogeneous graph neural network. In *SIGKDD*, pages 793–803, 2019.

[33] Y. Zheng, X. Xie, W.-Y. Ma, et al. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE DEB.*, 33(2):32–39, 2010.

[34] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. *arXiv preprint*, 2020.