

BasketballGAN: Generating Basketball Play Simulation Through Sketching

Hsin-Ying Hsieh

National Chiao Tung University
Hsinchu, Taiwan
x4ga71@gmail.com

Yu-Shuen Wang

National Chiao Tung University
Hsinchu, Taiwan
yushuen@cs.nctu.edu.tw

Chieh-Yu Chen

NVIDIA Corporation
Taipei, Taiwan
jaych@nvidia.com

Jung-Hong Chuang

National Chiao Tung University
Hsinchu, Taiwan
jhchuang@cs.nctu.edu.tw

ABSTRACT

We present a data-driven basketball set play simulation. Given an offensive set play sketch, our method simulates potential scenarios that may occur in the game. The simulation provides coaches and players with insights on how a given set play can be executed. To achieve the goal, we train a conditional adversarial network on NBA movement data to imitate the behaviors of how players move around the court through two major components: a generator that learns to generate natural player movements based on a latent noise and a user sketched set play; and a discriminator that is used to evaluate the realism of the basketball play. To improve the quality of simulation, we minimize 1.) a dribbler loss to prevent the ball from drifting away from the dribbler; 2.) a defender loss to prevent the dribbler from not being defended; 3.) a ball passing loss to ensure the straightness of passing trajectories; and 4.) an acceleration loss to minimize unnecessary players' movements. To evaluate our system, we objectively compared real and simulated basketball set plays. Besides, a subjective test was conducted to judge whether a set play was real or generated by our network. On average, the mean correct rates to the binary tests were 56.17 %. Experiment results and the evaluations demonstrated the effectiveness of our system. Code is available at <https://github.com/chychen/BasketballGAN>.

CCS CONCEPTS

• **Information systems** → **Multimedia content creation**; • **Human-centered computing** → **Interactive systems and tools**.

KEYWORDS

Conditional adversarial network, basketball, Sketch, Simulation

ACM Reference Format:

Hsin-Ying Hsieh, Chieh-Yu Chen, Yu-Shuen Wang, and Jung-Hong Chuang. 2019. BasketballGAN: Generating Basketball Play Simulation Through Sketching. In *Proceedings of the 27th ACM International Conference on Multimedia (MM '19)*, October 21–25, 2019, Nice, France. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3343031.3351050>

1 INTRODUCTION

Basketball is a popular sport that is played or watched by hundreds of millions of people in the world. It is a five against five players game, in which the goal is to score points by throwing the ball into the basket hoop. To prevent defensive stops by defenders, adopting tactics that can create an open space for the shooter is important. In practice, basketball coaches draw up players' movements and ball passing routes on a tactic board to show players how to execute an offensive tactic. After reading the sketched tactic, professional players would simulate the play based on their own intuition and experience, and then execute the tactic on the court as the coach drew up. However, this is not always the case for novice players. The play could differ a lot to the tactic expected by a coach due to the influence of defensive players on the court.

The main difficulty of executing an offensive tactic is not knowing how the opposing team will defend the tactic. Figure 1 is a simple ball rotation tactic designed by a coach, in which the goal is to confuse the defenders by moving the ball from top to bottom in a quick fashion. Because the defensive players' movements are unknown, it may not be easy to understand how and why the tactic is effective. Novice players would be less confident and dubious to the tactic when playing basketball games on the court. As a result, the flow of their offence has a higher probability to be disrupted by the opposing team. Even when the offense is performed by professional players, it is not guarantee that the tactic will succeed because the opposing team may react by adopting the right defensive tactic. Therefore, **a system that can simulate and predict how the opposing team will react would be very helpful to players and coaches**. Specifically, novice players can better understand the tactic and be more confident when they execute the tactic on the court. In addition, professional players and coaches can evaluate and determine whether an offensive tactic is effective when it is used against the opposing team. The simulation based on historical data can back up their intuition and help them make right decisions.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

MM '19, October 21–25, 2019, Nice, France

© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-6889-6/19/10...\$15.00

<https://doi.org/10.1145/3343031.3351050>

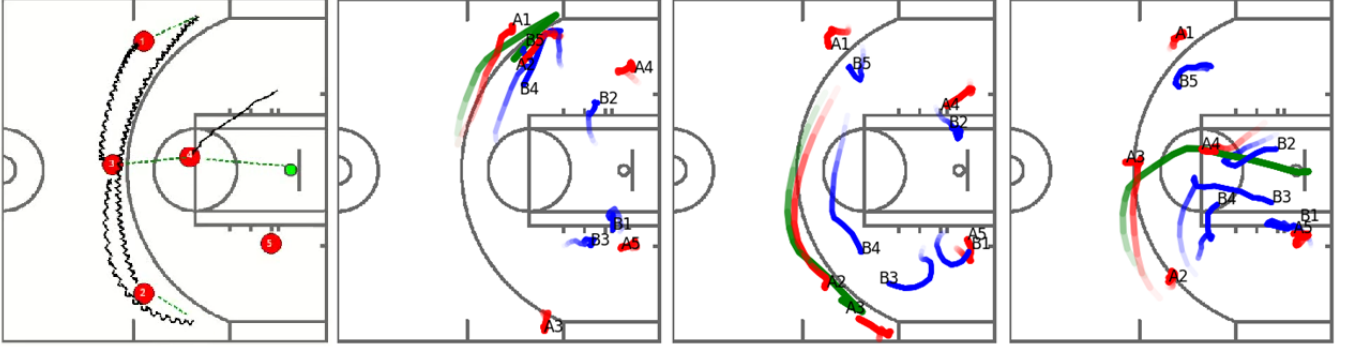


Figure 1: (Left) A sketched basketball offensive tactic. Green and red circles are the ball and the offensive players, respectively; zigzag, smooth, and dotted lines indicate the movements of the dribbler, non-dribblers, and the ball pass, respectively. **(Middle left to right)** The simulated basketball set play based on the sketched tactic. Offense, defense, and ball are coded in red, blue, and green, respectively. We segment the set play into three consecutive segments for clear visualization. Besides, trajectories from transparent to opaque indicate the movement direction. We recommend readers to watch our accompanying video for the sketching and the simulation results because the dynamic animations are difficult to visualize in still images.

Simulating a defensive play based on the offensive condition is not an innovative idea. Chen *et al.* [7] presented a generated adversarial network (GAN), which takes the ball and the offensive players’ movements as input, can achieve the goal. However, the system is infeasible because real movements of the ball and the players are difficult to sketch on a tactical board. Seidl *et al.* [27] transformed an offensive tactic sketch into an animation sequence and fed the sequence to a long short term memory network to simulate the defensive play. Since sketch lines are often smooth and lack temporal dynamics, the simulated plays are unrealistic – players move with an identical speed on the court. In addition, wide opens frequently occur in their results.

We present a GAN that is composed of convolution layers to simulate basketball set plays. The input of the network is an offensive tactic sketched by coaches and the output is a basketball set play, in which the offensive part fulfills the input condition. The presented GAN is composed of a generator and a discriminator. The former takes the condition and a random noise as input to generate basketball set plays, and the latter evaluates the realism of the plays. We train the network on a basketball player movement dataset released by NBA. The goal is to minimize the adversarial loss measured by the discriminator. In addition, to improve the realism of the simulation, we applied the dribbler loss, defender loss, ball passing loss, and the acceleration loss to guide the network training. These heuristic loss functions effectively prevent abnormal player behaviors on the court. Considering that users often draw smooth lines to specify an offensive tactic, and the lines lack temporal dynamics, we prepare the offensive conditions by spatially and temporally smoothing real players’ movement trajectories. The network is trained to construct details between sketches and real offensive players’ movement trajectories, and the missing defensive set play. After training, users can sketch an offensive tactic and forecast how the opposing team will defend the tactic.

We trained the network to simulate basketball set plays accordingly to the sketched offensive conditions. To evaluate the results, we compare real and simulated basketball set plays in terms of

the ball and the players’ movements, and their relationships. The results show that the basketball set plays simulated by our system are realistic. In addition, since quantitative evaluations may not be perfect, we also conducted subjective tests with 50 participants (6 professional players, 22 NBA fans, and 22 ordinary people) to evaluate the results. Participants were asked to identify the real play from two samples. On average, their mean correct rate to the binary tests were 56.17 %.

2 RELATED WORK

Basketball analytic has been a research trend for years. Early works attempt the use of statistic and number crunching in determining which player statistics are crucial and influence the outcome of a basketball game [17, 22, 25, 26]. Franks *et al.* presented a system [10] to determine which player is a better defender on specific locations on the court, and how efficient the player is at defending an offensive player. Beshai [4] implemented a system to visualize basketball shot data in the NBA. It provides users a platform to enjoy basketball statistics through interactive charts and player comparisons.

In the past few years, the existence of STATS SportVU player tracking data has opened new doors for researchers to examine players’ movements. Applications, such as tactic classification [6, 29] and shot make prediction [5, 14, 28] in basketball set plays, were presented by analyzing players’ movement trajectories. In addition, several works were introduced to generate basketball set plays. Toronto Raptors showcased a private system developed by their analytic team, which could use players attribute and skill-set to determine where each individual defensive player should stand on the court while on the defensive end. Seidl *et al.*[27] presented an interactive sketch play system. When given an input of offensive trajectories, their model can synthesize ghosting players that imitate defensive behaviors in basketball games. Chen *et al.* [7] trained a GAN that takes a real offensive play as input and generates the corresponding defensive play for coaches and players to consider. Although the works of Seidl *et al.*[27], Chen *et al.* [7]

and ours attempt to achieve a similar goal, their systems focus on only defensive plays. They either assume that offensive players move smoothly on the court or that realistic offensive set plays can be easily specified. In contrast, our system simulates realistic and complete basketball set plays according to simple sketches.

Generative adversarial network (GAN) is an unsupervised deep generative model pitting a generator and a discriminator against each other to generate an output $G(z)$ from a given latent noise z [12]. A GAN learns to generate realistic samples from a dataset distribution, as it has been used widely in content generation ranging from images, videos [18] to music [21] and text [9]. Because the input z is uncontrollable, a conditional version of GAN was then introduced [19]. By feeding a y condition into the network, we can specify a specific output rather than a generic sample generated from a random noise. Accordingly, many works have taken advantage of conditional GANs, such as generating images based on text description [23], style transfer of real image into a desired style [8] or image to image translation [15, 31].

Although GANs are promising and can achieve exceptional results, the network itself is difficult to train and experimentally suffers a few flaws such as models never converging or mode collapse [1]. To improve learning stability, Ajovsky *et al.* [2] presented Wassestein GAN (WGAN), which applies the Earth Moving distance to measure the similarity of real and fake samples. Later on, Gulragini *et al.* [13] improved upon the works of WGAN by gradient penalty and demonstrated the network is over its predecessor on quality and convergence. In addition to WGAN, there have been also several strategies presented to improve network training, such as energy-based GANs [30], minibatch discrimination [24], boundary equilibrium GANs [3], and spectral normalization [20].

3 BASKETBALL GAMES SIMULATION BY SKETCHING

We train a GAN to simulate basketball set plays according to the sketched offensive conditions. The network learns to construct details between sketches and real offensive players' movement trajectories, and the missing defensive set play. Details are described in this Section.

3.1 Training Data

The work dataset comes from Player Tracking data provided to the public from STATS SportVU. It contains NBA games during half of the regular 2015-2016 season (around 600 games played). The ball and the ten players' positions on the court were captured at 25 frame per second. We down-sample the tracking data to 5 frames per second for training. In addition, we segment a whole game into offensive plays that start when the ball is dribbled across or in-bounded from the half court, and ends when a shot is made or missed. To achieve said result, the NBA play-by-play information¹ is parsed to extract every shot made or missed, which can be done by matching the time of the play-by-play and tracking data since both datasets provide a time variable.

¹NBA play-by-play text information records every event type that occurs during a game by officials. Every event is a basketball play that occurs on the court, e.g. rebound, assist, shot made, shot attempted.

We prepare offensive conditions by imitating user sketches of those NBA player tracking dataset. The sketches are often smooth lines from a starting point to a desired end point, unlike real player movement trajectories, where players may move in unpredictable ways. In addition, the sketched movements in each segment are of uniform speeds because of the lack of temporal information. To synthesize sketched set plays, each real play is first segmented into segments by ball pass or ball shot at the basket, as these actions often indicate the end of a phase during a sketch play design. Then, each player movement trajectory is represented by a Bézier curve, in which the control points are simplified from the original tracking data by the Ramer-Douglas-Peucker (RDP) algorithm. The segment length is retained in this new representation. Different to players' movement trajectories, the ball trajectory is identical to the handler's trajectory when dribbled, and only moves in a straight line to an intended receiving player when passed, or to the basket hoop when shot.

3.2 Sketching Interface

We provide users with a graphical interface to sketch and design offensive set plays. The interface comes with a half-court basketball board, with five movable circles that represent the offensive players. Figure 1 (a) left shows an example. To sketch a play, users first set up the five players' positions, and then double click a circle to indicate the dribbler. By holding on a circle and moving it, the player's movement trajectory is sketched upon the board, displaying how the player moves from start to finish. Users can specify a ball pass by clicking the dribbler and then the intended receiver on the court. They also can create a shot in a similar fashion. Instead of an intended receiver, the sketch is directed towards the basket hoop.

The sketched set play lacks precise temporal information. Namely, the start and the end frame of a trajectory is unknown; and the player's movement speed is uniform. We consider the interfaces used in popular basketball apps and the work presented by Seidl *et al.* [27], and then decide to temporally segment an offensive play by ball pass and ball shot events. Each segment length (i.e., frame number) is determined by the longest trajectory and the mean player's movement speed, which is analyzed from the NBA player tracking data. In addition, we smooth the sketched trajectories in a similar fashion as the training data, and then determine the player position in each frame by uniform sampling. This approach can reduce the gap between real and synthesized sketches.

3.3 Generative Adversarial Network

We train a GAN to simulate basketball plays that can fulfill the offensive conditions sketched by users. Specifically, the input to the network is a set of sketch lines that condition the movements of a ball and five offensive players. The output is a complete basketball play that contains the movement trajectories of a ball, five offensive players, and additionally five defensive players. We recommend readers to watch the accompanying video for better understanding the input and output of the network because basketball plays are dynamic and difficult to visualize in still images.



Figure 2: Network Overview. Offense condition y and latent noise vector z are used to simulate a basketball set play $G(z|y)$. The simulated play is then evaluated by a critic network. The generator and the critic are trained to compete against each others.

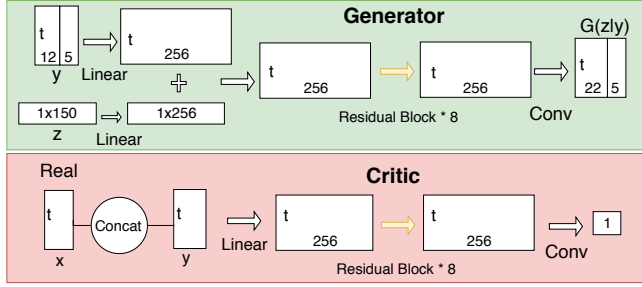


Figure 3: Details of the generator and the critic network architectures.

3.4 Network Architecture

Architecture overview. The presented GAN consists of a generator and a discriminator competing against each other, as illustrated in Figure 2. The generator G takes an offensive condition y and a latent noise vector z as input and generates the corresponding basketball play $G(z|y)$. Specifically, y is a $t \times 18$ matrix, where t is the frame number of a basketball play and each frame is conditioned by an 18 dimensional feature vector. Twelve dimensions of this vector are the ball and the offensive players’ positions \mathcal{R}^2 on the court. Six dimensions of the vector provide the ball status. When the ball is in possession of player or when a shot attempt occurs, a 1 is encoded to the corresponding player or the basket hoop. Otherwise, a 0 is encoded. Accordingly, at most one dimension of the ball feature can be 1; and a zero vector indicates a ball pass. The output of the generator $G(z|y)$ is a $t \times 28$ matrix, where t is the frame number the same to the input and each frame contains the positions of a ball and ten players, and a ball feature. By requesting the generator to generate its own ball feature, we can impact the generator to keep the ball near the handler when the ball is not passed or shot.

The discriminator evaluates the realism of a basketball set play $G(z|y)$. Specifically, it criticizes if the offense fulfills the given condition y , if both offence and defence together are realistic, and if the generated ball feature displays the correct ball status. Therefore, we form the real and fake pairs by $y \oplus x$ and $y \oplus G(z|y)$, respectively, to train the discriminator, where \oplus indicates concatenation.

Details of network architecture. Figure 3 shows the architectures of the presented generator G and the critic network C . In the generator, it is noting that the condition y and a latent noise z are linearly projected to have the same size before the feature maps are combined through addition. This enables the latent noise to have global influence over the entire sequence, rather than individual frames. The combined feature map is passed into 8 residual blocks

to construct the manifold feature. The output results from a down sampling convolution layer with stride 1 and a 5×5 kernel. In the critic, the real and fake pairs undergo the process similar to the generator. The network outputs a score to indicate the realism of a basketball play.

3.5 Loss Functions

We train the network by minimizing an adversarial loss, a dribbler loss, a defender loss, a ball passing loss, and an acceleration loss. The adversarial loss determines how real a generated basketball play is; the dribbler loss measures the distance between the ball and the dribbler; the defender loss penalizes the wide open; the ball passing loss ensures ball passing trajectories being straight lines; and finally, the acceleration loss prevents players from being too aggressive.

Adversarial loss. We compute the adversarial loss by measuring the Wasserstein distance [2] between real and fake distributions. The output of the critic is a scalar score that determines how real a sample is. Formally, the loss is written as:

$$\begin{aligned} L_g(G, C) = & E_{y \sim P_{data}(Y), z \sim P_z(Z)} [C(G(z|y)|y)] \\ & - E_{x, y \sim P_{data}(X, Y)} [C(x|y)] \\ & + \lambda \times E_{\hat{x} \in P_{data}(\hat{X})} E[(\|\nabla_{\hat{x}} C(\hat{x})\|_2 - 1)^2], \end{aligned} \quad (1)$$

where \hat{X} is the interpolation of the generated distribution and the real dataset, and λ is the weight for the regularization term. We set $\lambda = 10$ in the experiments.

Dribbler loss. The ball should not leave the dribbler when it is in possession. To prevent such a problem, we strive to minimize the Euclidean distance between the ball and the dribbler, where the dribbler is determined by the generated ball feature mentioned in Section 3.4. Let f_i be the ball feature that indicates whether player i possesses the ball, \mathbf{p}_b and \mathbf{p}_i be the ball position, and the position of player i , respectively, we minimize the loss

$$L_d = \sum_t \sum_i f_i \times |\mathbf{p}_b - \mathbf{p}_i| \quad (2)$$

in each frame of the generated basketball play $G(z|y)$, where t is the frame index, Note that $f_i = 1$ only if the player i possesses the ball, and that the loss will be 0 when the ball is passing or shooting. We show the results simulated by the network trained with and without this dribbler loss in Figure 4 (a).

Defender loss. A generated basketball play is unrealistic if the dribbler does not attempt to make scores when he/she is not defended. Hence, we minimize the Euclidean distance between the ball handler and the nearest defender that is between the player and the basket. To expect that the dribbler is defended as in the real basketball games, we minimize the following loss in each frame of the play:

$$\begin{aligned} L_w = & |D(x) - D(G(z|y))|, \quad \text{where} \\ D(\bullet) = & \sum_t (1 + \theta) \times (1 + |\mathbf{p}_n - \mathbf{p}_b|), \end{aligned} \quad (3)$$

x is a random real basketball play, \mathbf{p}_b is the ball position on the court, \mathbf{p}_n is the closest defender to \mathbf{p}_b , θ is the angle determined by $\mathbf{p}_n - \mathbf{p}_b$ and a vector from the ball to the basket. We show the

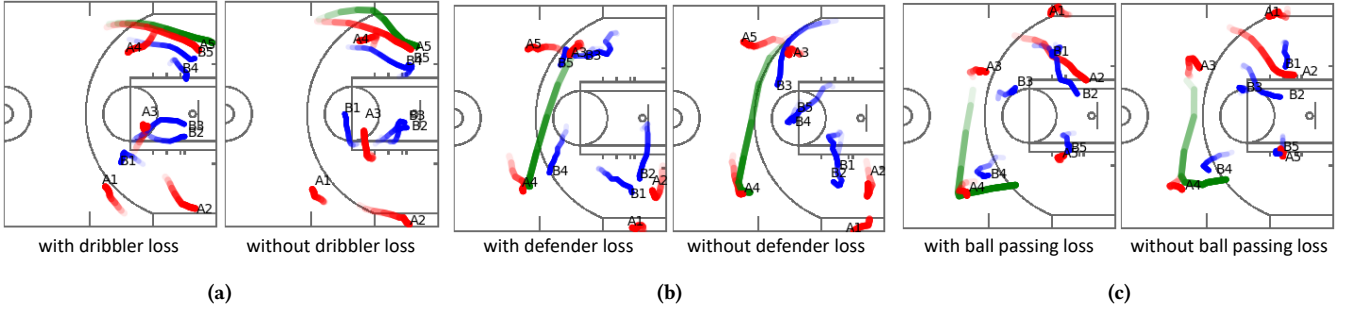


Figure 4: The basketball plays are simulated by our network that is trained with and without the dribbler loss, straight line loss, and the wide open loss. Green, red, and blue trajectories show the movements of the ball, offensive players, and defensive players, respectively. (a) In the top right region, the ball deviates from the ball handler. (b) In the bottom left region, a wide open occurs when the ball is passed from A3 to A4. (c) In the middle left region, the ball passing trajectory is not straight.

results simulated by the network trained with and without this defender loss in Figure 4 (b).

Ball passing loss. Simulated ball passing trajectories in the XY-plane should be straight. To fulfill this requirement, we add a straight line loss to guide the network training. Let $\mathbf{p}_{b,i}$ be the ball position at frame i , and ϕ_i be the angle formed by the ball positions $\mathbf{p}_{b,i-1}$, $\mathbf{p}_{b,i}$, and $\mathbf{p}_{b,i+1}$. The loss is formulated as

$$L_b = \sum_i \left(1 - \sum_i f_i \right) \times \phi_i. \quad (4)$$

Note that the loss is counted only when the ball is not in possession by players. We show the results simulated by the network trained with and without this ball passing loss in Figure 4 (c).

Acceleration loss. Professional basketball players would prevent unnecessary movements on the court to save their physical energies. However, simulated players may behave aggressively to fulfill the above-mentioned requirements, such as avoiding wide open. Therefore, to achieve realism, we add an acceleration loss to keep simulated players moving as real players. Specifically, the loss is defined as

$$L_a = |\mu(x) - \mu(G(z|y))|, \quad (5)$$

where $\mu(\bullet)$ is the mean acceleration of players in a set play.

By integrating the above mentioned losses, we minimize

$$L = L_g + w \cdot (L_d + L_b + L_w + L_a) \quad (6)$$

to update the network hyper-parameters, where $w = |C(G(z|y))|$ is the critic score that helps balance the overall influence of the adversarial loss and our defined losses.

3.6 Implementation Details

We trained the network on a single NVIDIA GeForce 1080ti GPU. Adam optimizer [16] with a learning rate $1e-4$ and a batch size 128 with a fixed sequence length ($t = 50$) were used. In addition, the hyper-parameters of the network were initialized using Xavier [11]. Although the network is trained with a fixed sequence length, during testing phase variable length sequences can be fed into the network for basketball play simulation.

A three step strategy is implemented for obtaining the best results. First, the critic is pre-trained for 10 epochs with generator

only trained for 1 iteration per epoch. This ensures that the critic is optimum to start with and feed meaningful gradients to the generator to improve upon. Then, we train the critic for 5 iterations before training the generator once, and with every 20 epochs we update the critic 10 iterations per generator iteration, as this strategy keeps the critic remaining strong. We stop the training process when the critics start overfitting.

4 RESULTS AND EVALUATIONS

We tested the trained neural network on a variety of sketched offensive conditions, including those consisting of frequent ball passing and screening. After the ball and the offensive players' movement trajectories are sketched on the tactical board, our system will simulate the basketball set play. Figure 5 shows several examples simulated by using our system. The offensive and defensive players are termed as A1-A5, and B1-B5, respectively. Considering that visualizing basketball set plays by using still images is difficult, we recommend readers to watch our supplemental videos for a better understanding of the plays.

Snap down. The set play involves getting the attention away from A1 as he cuts away from the play after passing the ball to A5. After the initial hand-off, a down screen is set for A1 by A4, where A1 gets free from his defender. Meanwhile, A5 and A3 pass back and forth once, before the ball is passed to A4 after the screen is set. Then, A5 proceeds to set screen for A1 to get an open shot, receiving a pass at the top of the key from A4.

Hammer. This example shows how to get A5 open. In the beginning, A5 set the Hammer screen (off the ball screen) for A2 on the weak side. Typically, A5 is a center and not known for hitting shots outside the restricted area. This leads to A5 defender (B5) to be prone to sag off and help either defending A2 when the screen is set, or protect the rim. This helps opens up for a jump shot or cut into the restricted area for an easy layup from a pass by A4.

Flex. A flex cut occurs when a player in the corner receives a screen from another player that leads him into the restricted area for an easy layup or shot. In this example, two separate flex cuts occur, starting with A3 receiving a screen from A5. The first flex is a dummy, as A5 moves up to receive the ball. A3 then screens for A4 to flex cut into the restricted area for an easy layup.

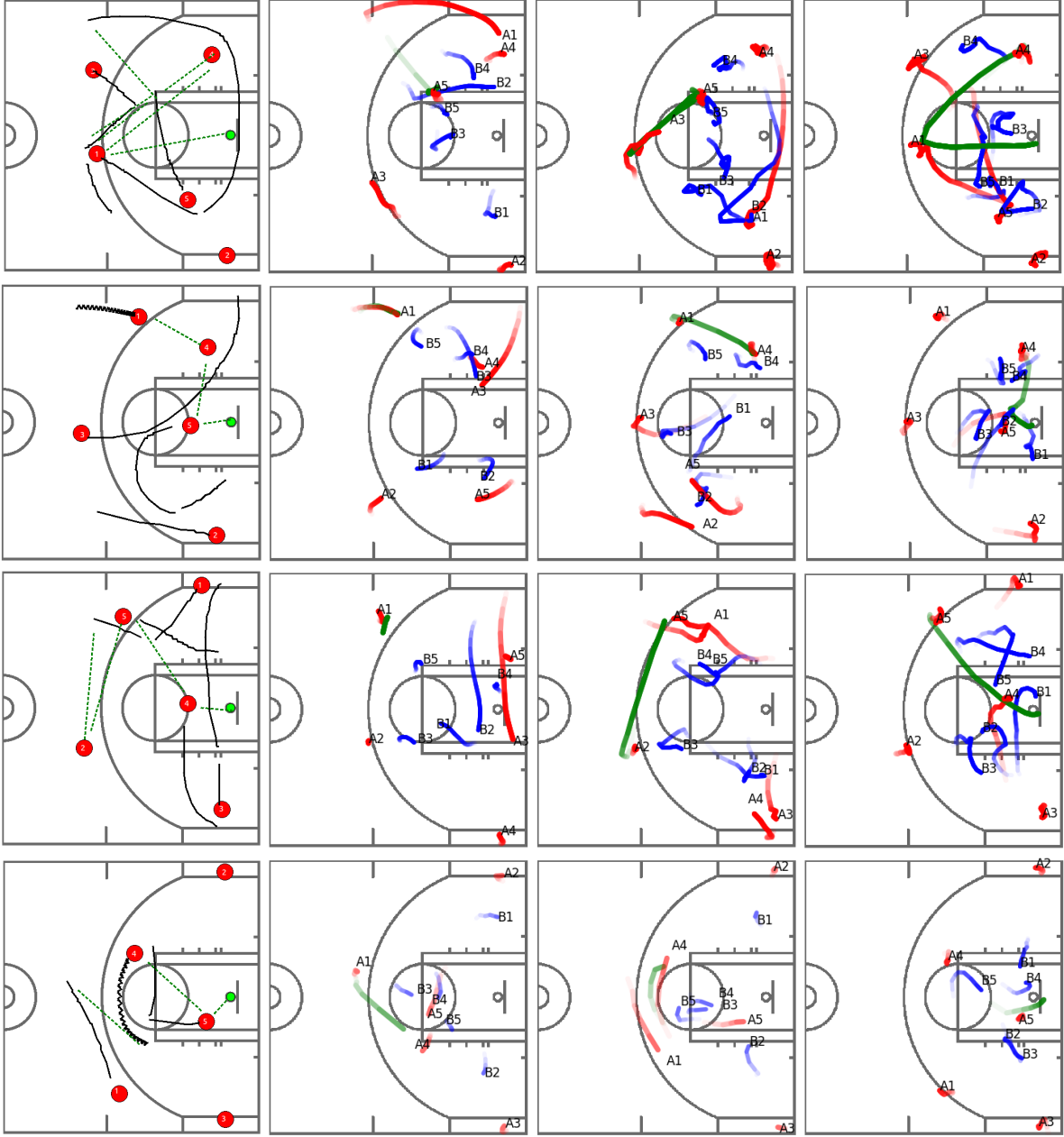


Figure 5: We show basketball set plays simulated by our system based on four different offensive conditions. The tactics from top to bottom are snap down, hammer, flex, and elbow. The first column shows the sketches. The second to the fourth columns are the consecutive segments of our simulated results.

Elbow. The offence is performed by two big men (center and power-forward) who utilize the pick-and-roll action to create an open. After A1 passes the ball to A4, A5 sets a screen for A4 on top of the key. Once the screen is set, A5 rolls to the basket, giving the options for A4 to shoot or pass the ball to A5 for an easy layup. As shown in this this example, an open is created after A4 receives 3 defenders attention.

4.1 Data Order

Players' positions in a condition y and the simulated basketball set play $G(z|y)$ are represented by a $t \times 10$ and a $t \times 20$ matrices, respectively. The order of these players strongly affects the convergence speed. Specifically, if the first player acts as a guard in a set play and as a forward in another, the network had to take more steps to learn the simulation of basketball set plays. To speed up the training

	Ball		Offensive players		Defensive players		Ball-player relationship	
	velocity	acceleration	velocity	acceleration	velocity	acceleration	Ball-dribbler Dis	Ball-defender Dis
Real	9.52 (8.62)	4.59 (5.47)	5.01 (3.78)	0.85 (0.94)	4.03 (2.93)	0.83 (0.92)	1.55 (0.79)	6.37 (2.75)
Ours	7.43 (6.98)	2.29 (3.38)	4.93 (3.48)	1.17 (1.31)	4.12 (2.87)	1.19 (1.36)	1.53 (1.56)	6.17 (3.03)
C-VAE	7.65 (7.10)	2.60 (3.25)	4.68 (3.35)	1.12 (1.37)	3.22 (2.16)	0.97 (1.22)	1.47 (1.43)	6.09 (2.49)
DEF	x	x	x	x	4.29 (3.50)	1.45 (2.18)	x	6.05 (2.29)

Table 1: Statistics of real and simulated basketball set plays. The 2rd to the 7th columns show the mean velocity and acceleration of the ball, offensive players, and defensive players, respectively. The 8th and 9th columns show the distances from the ball to the dribbler and to the closest defender, respectively. The number in each bracket indicates the standard deviation. We highlight the statistic of the simulated data closest to that of the real data to facilitate comparison.

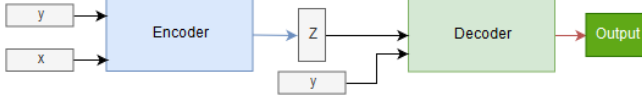


Figure 6: The architecture of C-VAE. During the training phase, both the condition y and the real basketball set play x are required. During the testing phase, the encoder can be neglected. Only the condition y and a latent noise z are fed into the decoder for reconstruction.

process, we sort offensive players according to the mean distances between the players and the ball in a set play. Then, each defensive player was attached to the closest offensive player to obtain his order. This sorting strategy enables the players playing the same position to be likely arranged in the same column of the matrix. We realize that this sorting method was not optimal because players' movements on the court are complex. However, experimentally we observed that the overall training steps can be reduced by more than 2 times by the sorting.

4.2 Model Comparison

For quality evaluation purposes, two other generative models along with our system were trained and were compared with each other. Since conditional variational autoencoder (Figure 6) is known for good data reconstruction, we compared our system to this C-VAE network. In addition, we compared our system to the work of Chen *et al.* [7], which takes real offensive set plays as input and outputs the corresponding defensive set plays. In the comparison, the input to this DEF network was the same to the other networks. The results of the comparison are shown in our supplemental material.

Table 1 shows the statistic of real and simulated basketball set plays that were experimented on the testing dataset. The ball, offensive players, and defensive players movements were evaluated. As indicated, our system performs better than C-VAE and DEF on players' movement velocity and the relationships between the ball and the related players. However, it makes the simulated players become aggressive as their accelerations are higher than the players simulated by other methods. We suspect the reason was that the discriminator determined the realism of a basketball set play by examining a sequence of player positions. The second derivatives of the player positions were considered less important. From the human perspective, this strategy is not harmful because the change of acceleration is less noticeable than the change of speed

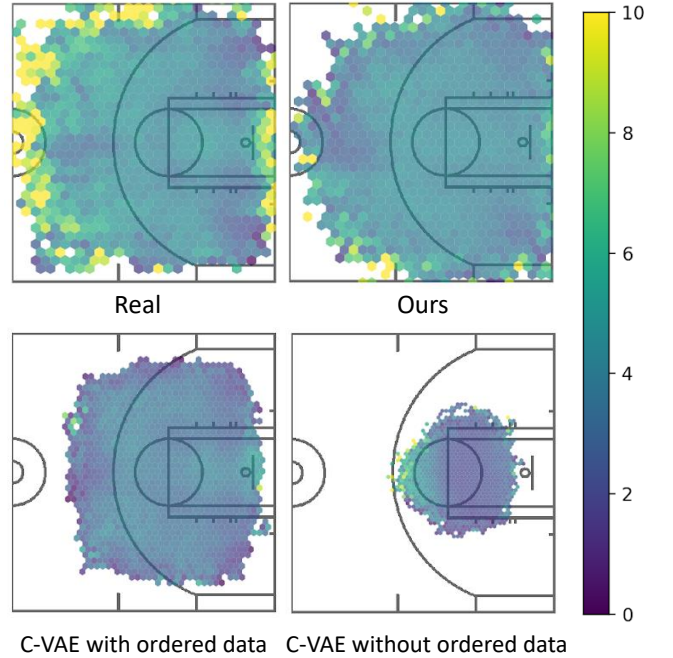


Figure 7: We show the distributions of defensive players on the court. The mean players' velocity at each position is represented by color. On the right is the transfer function.

and position to humans. In addition, we observed that C-VAE could successfully reconstruct movements of the ball and the offensive players. The result was not surprising because of the hints from the given condition y . However, their simulated defensive players moved quite slowly on the court and tended to stay in two-point zone, as visualized in Figure 7. The situation occurred because C-VAE is a supervised learning network. Finally, DEF was trained to simulate only defensive players. Besides the unrealistic offensive set plays, the simulated defensive set plays were also less realistic than ours.

It deserves nothing that the order of players' data strongly influenced the results of C-VAE. When the order was randomly assigned, the mean velocity of the defensive players dropped from 3.22 to 2.15. In addition, the players got closer to the restricted area (Figure

	Section 1: true or false						Section 2: left or right					
	Q1	Q2	Q3	Q4	Q5	Q6	Q1	Q2	Q3	Q4	Q5	Q6
ORD	0.32	0.64	0.59	0.50	0.50	0.55	0.41	0.50	0.55	0.50	0.50	0.55
FAN	0.18	0.86	0.59	0.55	0.73	0.59	0.68	0.73	0.40	0.31	0.64	0.59
PRO	0.33	0.83	0.83	0.33	1.00	0.67	0.83	0.83	1.00	0.50	0.67	0.83

Table 2: We show the mean correct rates of each question answered by three different groups.

7). By contrast, the GAN-based models did not suffer from the problem. In our experiments, although the training steps considerably increased under this circumstance, the quality of the simulations was held.

4.3 User Study

Conducting an objective measurement is a good way to evaluate the quality of the simulated basketball set plays. However, with subjective test results, one can be more convinced that our simulated basketball set plays resemble real natural movements. Therefore, we created a survey that consists of two sections, each with 6 questions. The simulated results were selected from the testing dataset. In section 1, participants were faced with 3 real and 3 fake basketball set plays and were asked to answer whether the play was real or fake. The real and fake plays were displayed in a random order to avoid bias. In section 2, both real and fake plays were compared side by side. Participants had to determine the play on the left or on the right was real. Similarly, the position of each play was randomly determined. All 50 participants that took the survey have experience in activities of watching and playing basketball. Participants were categorized into three groups, ORD, FAN and, PRO. ORD were participants who know basketball rules and seldom watch basketball games; FAN were participants who often watch NBA TV, and PRO were participants who joined clubs and actively played basketball.

Table 2 shows the mean correct rate of each question. Overall, the mean correct rate to all of the binary tests were 56.17 %. Among the three groups, PRO performed the best. 67% and 78% of the questions in Section 1 and 2 were correctly answered, respectively. Their answers to question 5 in section 1 and question 3 in section 2 were all correct. FAN was the second. 58% and 56% of the questions in Section 1 and 2 were correctly answered, respectively. ORD was the last. 51% and 50% of the questions in Section 1 and 2 were correctly answered, respectively. Their answers were close to random guess. In addition, the overall statistic was reasonable, as PRO was the group the most familiar with tactics and ORD was the least. We also observed that PRO could correctly answer more questions in Section 2 than in Section 1. While real and fake basketball plays were put together and compared side by side, they were likely to pick the right play.

4.4 Limitations

We present a network to simulate realistic basketball set plays according to the sketched offensive conditions. A noted limitation is that input sketches for training are simplified from real players' movement trajectories rather than being specified by users. Although experimentally we did not observe noticeable artifacts in the simulations based on user sketches, theoretically we cannot

claim that the two types of sketches are the same. In addition, we observed that offensive players not given instructions will not leave their positioned region in the simulation. However, in reality the players react on their teammates actions. They may help create a wide open or receive the ball to make scores. On the other hand, defensive players were simulated without receiving any instructions. One idea to improve the system is conditioning defensive schemes by labels such as zone defence and man-to-man. As a result, players and coaches will be able to forecast different defensive reactions to their offense. Finally, the system imitates just general player movements when it synthesizes basketball set plays. In practice, it is beneficial for coaches to teach their players. In competition, however, the system can be improved further to imitate a higher-level of specific team or player movement behavior. By taking account of players' profiles, we could see different defensive approaches such as not pressuring a poor three-point shooter when he has the ball around the three-point line.

5 CONCLUSIONS AND FUTURE WORKS

We have presented a deep neural network to simulate basketball set plays according to sketched offensive conditions. The network is trained on NBA player tracking data and learns to construct details between sketches and real offensive players' movement trajectories, and the missing defensive set plays. To achieve realistic simulations, we introduced several loss functions to assist network training. Experiment results and comparisons demonstrated that simulated basketball set plays can be indistinguishable from real set plays. Accordingly, by using our system, players and coaches can rely on objective data analysis rather than subjective intuition, and foresee how the defensive team will react to their offensive tactic. The simulation of basketball set plays help players get insights from tactics and help coaches make crucial decisions.

Our current network is trained on general basketball players. It can be fine-tuned to simulate basketball set plays that a specific team or player is likely to react. In this case, the network will be particularly useful for coaches to design tactics and select players as the starting lineup against the opposing team. We thus plan to collect and prepare data for network fine tuning in the near future. We will also improve the usability of our system and release the codes/program for public use.

ACKNOWLEDGEMENTS

We thank anonymous reviewers for their insightful comments and suggestions. We are also grateful to Prof. Chin-Jen Cheng for the valuable discussions, and all the participants who joined the user study. This work is partially supported by the Ministry of Science and Technology, Taiwan, under Grant No. 105-2221-E-009 -135 -MY3 and 107-2221-E-009 -131 -MY3.

REFERENCES

- [1] Martin Arjovsky and Léon Bottou. 2017. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862* (2017).
- [2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875* (2017).
- [3] David Berthelot, Thomas Schumm, and Luke Metz. 2017. BEGAN: boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717* (2017).
- [4] Peter Beshai. 2014. Buckets: Basketball Shot Visualization. *University of British Columbia, published Dec* (2014), 547–14.
- [5] Daniel Cervone, Alex D’ÁZAmour, Luke Bornn, and Kirk Goldsberry. 2016. A multiresolution stochastic process model for predicting basketball possession outcomes. *J. Amer. Statist. Assoc.* 111, 514 (2016), 585–599.
- [6] Ching-Hang Chen, Tyng-Luh Liu, Yu-Shuen Wang, Hung-Kuo Chu, Nick C Tang, and Hong-Yuan Mark Liao. 2015. Spatio-Temporal Learning of Basketball Offensive Strategies. In *Proceedings of ACM international conference on Multimedia*. 1123–1126.
- [7] Chieh-Yu Chen, Wenze Lai, Hsin-Ying Hsieh, Wen-Hao Zheng, Yu-Shuen Wang, and Jung-Hong Chuang. 2018. Generating Defensive Plays in Basketball Games. In *2018 ACM Multimedia Conference on Multimedia Conference*. ACM, 1580–1588.
- [8] Yang Chen, Yu-Kun Lai, and Yong-Jin Liu. 2018. CartoonGAN: Generative Adversarial Networks for Photo Cartoonization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9465–9474.
- [9] William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: Better text generation via filling in the . *arXiv preprint arXiv:1801.07736* (2018).
- [10] Alexander Franks, Andrew Miller, Luke Bornn, and Kirk Goldsberry. 2015. Counterpoints: Advanced defensive metrics for nba basketball. In *9th Annual MIT Sloan Sports Analytics Conference, Boston, MA*.
- [11] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 249–256.
- [12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.
- [13] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*. 5767–5777.
- [14] Mark Harmon, Patrick Lucey, and Diego Klabjan. 2016. Predicting Shot Making in Basketball Learnt from Adversarial Multiagent Trajectories. *arXiv preprint arXiv:1609.04849* (2016).
- [15] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. *arXiv preprint* (2017).
- [16] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- [17] Justin Kubatko, Dean Oliver, Kevin Pelton, and Dan T Rosenbaum. 2007. A starting point for analyzing basketball statistics. *Journal of Quantitative Analysis in Sports* 3, 3 (2007).
- [18] Yitong Li, Martin Renqiang Min, Dinghan Shen, David Carlson, and Lawrence Carin. 2018. Video Generation from Text. (2018).
- [19] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).
- [20] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral Normalization for Generative Adversarial Networks. In *ICLR*.
- [21] Olof Mogren. 2016. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904* (2016).
- [22] Dean Oliver. 2004. *Basketball on paper: rules and tools for performance analysis*. Potomac Books, Inc.
- [23] Scott Reed, Aäron van den Oord, Nal Kalchbrenner, Victor Bapst, Matt Botvinick, and Nando de Freitas. 2016. Generating interpretable images with controllable structure. (2016).
- [24] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*. 2234–2242.
- [25] Jaime Sampaio, Eric J Drinkwater, and Nuno M Leite. 2010. Effects of season period, team quality, and playing time on basketball players’ game-related statistics. *European Journal of Sport Science* 10, 2 (2010), 141–149.
- [26] Jaime Sampaio, Manuel Janeira, Sergio Ibáñez, and Alberto Lorenzo. 2006. Discriminant analysis of game-related statistics between basketball guards, forwards and centres in three professional leagues. *European journal of sport science* 6, 3 (2006), 173–178.
- [27] Thomas Seidl, Aditya Cherukumudi, Andrew Hartnett, Peter Carr, and Patrick Lucey. 2018. Bhostgusters: Realtime Interactive Play Sketching with Synthesized NBA Defenses. (2018).
- [28] Rajiv Shah and Rob Romijnders. 2016. Applying deep learning to basketball trajectories. *arXiv preprint arXiv:1608.03793* (2016).
- [29] Kuan-Chieh Wang and Richard Zemel. 2016. Classifying NBA offensive plays using neural networks. In *Proc. of MIT Sloan Sports Analytics Conference*.
- [30] Junbo Zhao, Michael Mathieu, and Yann LeCun. 2016. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126* (2016).
- [31] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint* (2017).