

Classifying NBA Offensive Plays Using Neural Networks

Kuan-Chieh Wang, Richard Zemel
University of Toronto
Toronto, Ontario, Canada
wangkua1@cs.toronto.edu, zemel@cs.toronto.edu

Abstract

The amount of raw information available for basketball analytics has been given a great boost with the availability of player tracking data. This facilitates detailed analyses of player movement patterns. In this paper, we focus on the difficult problem of offensive playcall classification. While outstanding individual players are crucial for the success of a team, the strategies that a team can execute and their understanding of the opposing team's strategies also greatly influence game outcomes. These strategies often involve complex interactions between players. We apply techniques from machine learning to directly process SportVU tracking data, specifically variants of neural networks. Our system can label as many sequences with the correct playcall given roughly double the data a human expert needs with high precision, but takes only a fraction of the time. We also show that the system can achieve good recognition rates when trained on one season and tested on the next.

1. Introduction

An untrained spectator would have difficulty recognizing the many different offensive play calls in an NBA game. While the slam dunks, long-range 3-pointers, and buzzer-beaters make the highlight reels, what leads to these exciting events and victories are often complicated movements and interactions between all five players. Traditionally, human annotators such as assistant coaches or scouts recognize these plays in real-time by reading players' movements and the signaling gestures on and off the court. Annotating these plays provides useful scouting information, and also helps a team evaluate its own strategy. However, offensive strategies are complex and dynamic, and can be executed with multiple variations as the defense reacts. Unlike recording a shot outcome, a rebound, or other simple events, this task requires a high-level understanding of the game. Automating the process of detecting plays in video would not only reduce the considerable work involved but could also give rise to more detailed scouting reports, and other advanced statistics.

In this paper, we tackle the challenging problem of classifying NBA offensive playcalls directly from SportVU's tracking data. We use deep neural networks to build classifiers that can recognize plays, learning based on a small training set of annotated plays. We take two further innovative steps. First, unlike traditional analytics methods that depend on static location information, we analyze the stream of location information over time, employing state-of-the-art recurrent neural network (RNN) models. We show that this leads to significant improvements in play recognition. Second, we evaluate our system in the scenario of transfer to a new season. Given the models developed using 2013-2014 season's data, we study how well the algorithms help us classify plays from the beginning of the 2014-2015 season.

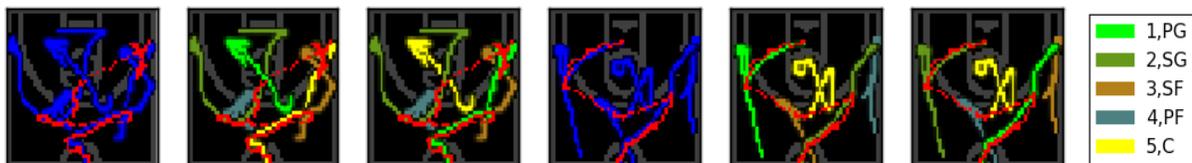


Figure 1: Pictorial Representation. 2 sets of 3 images from 2 different instances of the same playcall. The thinner red trajectory is the ball. The leftmost of the 3 is single-colored representation for all 5 players. The middle is players colored by the order of sequences in SportVU raw data. The rightmost is colored after we disambiguate on court position. Looking at the position-ordered images, both instances the PG initiated from halfcourt while the same positions occupy similar regions on the court, but in the raw SportVU data we see a random ordering of player positions.

From a machine learning perspective, this task is most similar to that of action recognition. Typically, models are developed to watch video such as movies or Youtube clips, and classify what type of video clip it is (e.g., a romantic movie, or a tennis match) [1, 2]. However, these approaches typically employ very complicated computer vision algorithms, specifically crafted for the given set of videos. Here we have the advantage of using the SportVU tracking data which contains all the players' coordinates on the court. Even though players' appearance could add information useful to deciding what play it is, most playcalls can be identified with just the evolution of their coordinates. Experiments with human experts showed that on 100 test sequences, unless the play breaks down due to time-outs, fouls, or ended prematurely, looking at moving dots from SportVU data representing player positions was enough for nearly perfect classification.

2. Method

2.1. Pictorial representation of player location sequences

SportVU data contains each player's coordinates on the court and the 3D position of the ball at 25 frames per second. Along with the coordinates it provides a unique player identifier. However, (x,y) coordinates are not sensible representations for our model. We are trying to learn a mapping between the input sequence and the type of play. (10, 10) is twice of the value of (5,5) in the number system, but that does not mean (10,10) contains 2 times the signal for a certain playcall than that of (5,5). We therefore adopt a pictorial representation of the player and ball positions, where each position is a small circle in the image. To simplify our input representation further, we combined player position sequences to form a single image containing player 'foot-prints' throughout the play (see Figure 1). This converts the problem into one of image classification.

2.2. Standard neural network (NN) for play classification

Neural networks refers to a family of algorithms from the machine learning community, which has attracted a lot of attention over the last decade [3]. They have broken records in many challenges that researchers have worked on for decades, in fields like computer vision and speech recognition. For the sports analytics community, it is beneficial to use neural networks because they are relatively easy to apply to difficult problems, and can be easily maintained.

A key question for any system is the representation of the input. One option is to utilize features extracted from the SportVU data that are useful for play recognition. These features could encode regular occurrences at different locations on the court (e.g., point guards tend to stand at the key

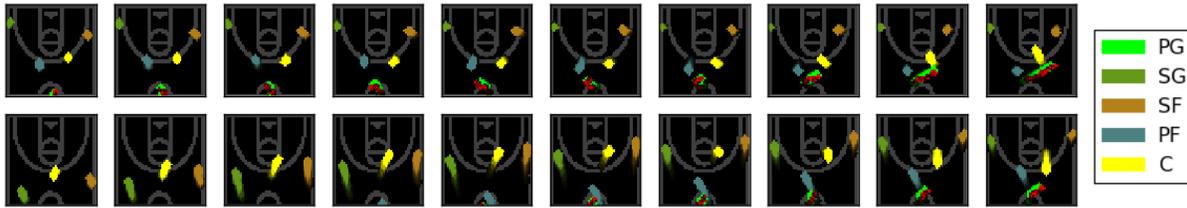


Figure 2: Sequential representation. Time starts from left and evolves for 4 seconds at 2.5 frames per second towards right. They are different instances of a common strategy named ‘horns’. While the 1st row we can identify the ‘horns’ set up pretty earlier (from frame 1), the second row took nearly 4 seconds to reach that formation.

while centers are in the paint), and also frequent patterns such as screens, cuts, drives, and pick-n-rolls. These features are difficult to define and extract. Neural networks, on the other hand, do not depend on hand-coded, engineered representations of input in terms of features, but instead allow these features to be learned from the data. In this paper, neural network (NN) models take as input the pictorial representation of SportVU data as described above.

2.3. Temporal information and recurrent neural networks (RNN)

As observed in Figure 1, the foot-print images are difficult to understand because the time information of how players move relative to each other and the ball is lost. We therefore separate the image into time steps, to encode where players are at each time step. At each step the input is the current pictorial representation of the player position. We also include some fainter representation of player positions from the previous frames, visually similar to the shadow of a player (See Figure 2). Further, since our models learn by examples, due to the limited number of labelled sequences they could just memorize certain coordinates or timeframes that are not necessarily representative of the playcall. To prevent a model from memorizing those uninformative features, we also add uncorrelated white noise to each frame in the sequential input. A recurrent neural network (RNN) is a variant of neural networks that can deal with sequential data of variable length (see Figure 3). The algorithm itself is stationary across time; only the evidence it accumulates change over time. Intuitively, players are governed by the same set of dynamics no matter when it is, but they could be at different locations doing different things given their previous movements.

For the RNN, our problem can be described as mapping a sequence $\mathbf{x} = (x_1, \dots, x_T)$, which contains a representation of the SportVU coordinates from time 1 to T, to a class label y , indicating which of the K strategies it is. This mapping is done through a series of weights. For a typical RNN as shown in Figure 1, the hidden unit at time t receives inputs from x_t below and h_{t-1} which is the evidence accumulated as far through weight matrices \mathbf{W}_{in} and \mathbf{U} respectively.

$$h_t = \sigma(W_{in}x_t + Uh_{t-1} + b_{in})$$

where σ is a non-linear activation function such as a sigmoid, or rectified linear unit (ReLU).

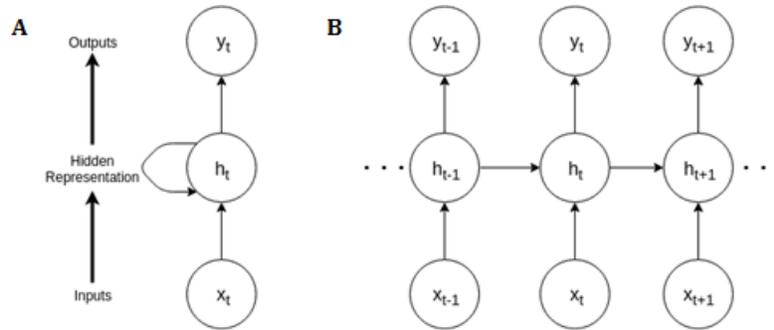


Figure 3: Recurrent neural network architecture. The inputs correspond to the player coordinates from SportVU, and outputs are the class labels for the different offensive strategies. Diagram A shows what the model really looks like where the hidden connections feed to itself. Diagram B illustrates what the network looks like rolled out in time. For simplicity of the illustration, there is only 1 hidden layer, but in practice it could be of variable depth.

The RNN learns to model $\Pr(y|\mathbf{x})$ the conditional distribution over the class labels given the input sequence.

$$\Pr(y_t|x_1, \dots, x_t) = \text{softmax}(W_{out}h_t + b_{out})$$

Learning for a RNN means finding a set of parameters $\theta = \{W_{in}, W_{out}, b_{in}, b_{out}, U\}$ that optimizes this distribution given the training data $\{(\mathbf{x}^1, \mathbf{y}^1), \dots, (\mathbf{x}^N, \mathbf{y}^N)\}$.

$$Cost_{class} = \prod_{n=1}^N \Pr(y^n | \mathbf{x}^n)$$

$$\theta = \text{argmax}_{\theta} Cost_{class}$$

In this study, we use a popular variant of RNN with long short-term memory (LSTM) units [4].

2.4. Ordering the sequences and player embedding

Another confounding factor is the representation of the input in terms of player identifiers. The same play can be executed with many different players. Basketball plays are typically designed based on the player positions. We therefore need to infer the position of each player on the court. The problem of converting a player identifier to his position is complicated, due to the fact that depending on who is on the court, a player could change his role.

In order to disambiguate player position given any line-up, we trained an autoencoder neural network [3] based on a player's shooting tendencies to map each player to a 10-dimensional embedding space where neighbors in this space have similar tendencies (e.g., range, movements prior to the shot, etc.). We then resolve a player's position based on his coordinates and neighbors in the embedding space (see Figure 4).

In Table 1 we show an example where this approach successfully resolves players' positions, taking into account the full set of players on the floor.



PG	SG	SF	PF	C
Kyle Lowry	Lou Williams	Terrence Ross	DeMar DeRozan	Jonas Valanciunas
Kyle Lowry	Lou Williams	DeMar DeRozan	Amir Johnson	Jonas Valanciunas
Lou Williams	Terrence Ross	DeMar DeRozan	Amir Johnson	Jonas Valanciunas

Table 1: Example of player position resolution. Here are 3 possible line-ups of the Toronto Raptors where Lou Williams, Terrence Ross, and DeMar DeRozan were assigned different positions depending on the line-up.



Figure 4: A visualization of player embedding. This show Lou Williams, Terrence Ross, and DeMar DeRozan, and their neighbors in the embedding. This is a 2D compression of the 10D space, so the distance in the visualization does not have the same semantics as the real embedding. Still, we see that Lou Williams is closer to other smaller guards while DeMar DeRozan is closer to bigger guards, and Terrence being somewhere in between.

2.5. Incorporating sequence prediction

With a limited number of examples of each play in the training set, it is possible for a model to perform well on play classification without acquiring an understanding of player movements. A model that does not understand movement may not generalize to unseen examples well. We can explicitly test the model’s understanding of movements by asking it to predict where the players will be in the next frame; we call this task *sequence prediction*. A second motivation for sequence prediction is the abundance of unlabelled plays: the advantage of this task versus play classification is that we have as many labels as we have input data.

Since our end goal is still play classification, we formulate our model to share a common encoding structure (i.e. from x_t to h_t in Figure 1), which is then fed into two decoders: one for the play classification task as formulated before, another for sequence prediction. Given sequence $\mathbf{x} = (x_1, \dots, x_t)$, the model is a function F that predicts \hat{x}_{t+1} which is the (x,y) coordinate for the $t+1^{\text{th}}$ timestep:

$$\hat{x}_{t+1} = F(x_1, \dots, x_t)$$

The objective is to minimize the Euclidean distance between the prediction and ground truth.

$$Cost_{seq} = \sum_{t=k+1}^T \sum_{n=1}^N \|x_t^n - \hat{x}_t^n\|_2$$

where k is the frame that we want to model to start sequence prediction. In this study we let the model start sequence prediction after 2 seconds from the start frame. Now when we find our best model, we optimize both objectives jointly:

$$\theta = \operatorname{argmax}_{\theta} (w * Cost_{seq} + Cost_{class})$$

where w is a scalar weight to control how much emphasis we place on sequence prediction.

2.6. ‘Anytime’ prediction

Basketball is a dynamic sport. Even though the shot-clock is always 24-seconds in the NBA, a play can start at any time. The same play may take a varying amount of time to set up and execute,

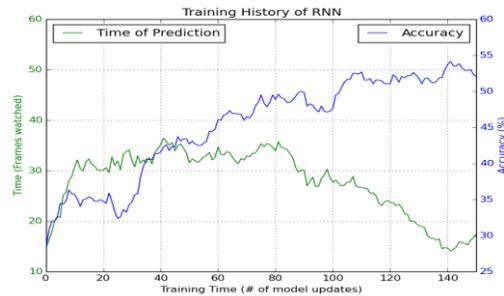


Figure 5: Training history of RNN. This figure show that upon random initialization, the model is not accurate. Thus at the beginning of training, it delays its prediction to maximize the accumulated evidence. As learning progresses, it requires less evidence and time to make good predictions.

depending on teammates' readiness and defensive pressure. Also, for different strategies, the distinguishing aspect can occur at different times. For example, while the play HORNS is easily spotted by its initial set-up, other plays can only be disambiguated by the last few frames. Hence, for the RNN, a natural solution is to allow prediction at any frame (see Figure 5).

3. Experiments

The data used for this study was drawn primarily from the SportVU dataset for the regular season games from the 2013-2014 season. The labeled play dataset was provided by the Toronto Raptors, for the offensive strategies used in that season. A total of 11 play classes were selected. The original number of play sequences was 7481, but after filtering down for the 11 classes, we were left with 1435 sequences. Many of the discarded sequences were transition offense, after timeout, in-bound plays and other less frequent plays. After timeout and in-bound plays both have distinct markers in the SportVU raw data, hence it is reasonable to segregate them from normal half-court plays. Transition offense is fairly easy to identify, and its nature differs from other half-court strategies quite drastically, so both technically and conceptually there are good reasons to dismiss them.

We kept 95 labelled sequences as an unseen test set. The remaining 1340 examples were split into 10 equal batches for training and validation. All of these batches and the test batch were balanced for their class distribution after randomization.

In Section 3.1, we will first report the performance characteristics of the models on the dataset described. In Section 3.2, we explore an interesting real-world setting where the model trained on the 2013-2014 dataset is used to make predictions about the 2014-2015 dataset.

3.1. Classification performance

In order to assess our approach, we evaluate the different models based on several metrics for evaluation. The first metric is top-1 accuracy, which compares the single highest-scoring class by the model to the correct answer, on each test example. Top-k accuracy considers the k classes that attain the highest scores for the model on that example. Another valuable metric is precision and recall, which depends on a threshold; if the score of the model's highest scoring class is below the threshold the model outputs "don't-know" for that example.

We examine three different models. A baseline is the most naive approach, which simply guesses the same class for each example, which is the most likely class in the training set. For top-k



Models	top-1 accuracy	top-3 accuracy	Precision/ Recall at T=.4	Precision/ Recall at T=.7
base-rate	.137	.390	n/a	n/a
NN	.547	.779	.724/.412	.909/.196
RNN	.656	.806	.727/.918	.900/.590

Table 2: Classification performance.

accuracy, we just predicted based on the top-k most likely classes in our training set. For our 11-class dataset, the top-1 base-rate was 13.7% in the test set.

A neural net (NN) trained on the pictorial representation achieved 54.7% top-1 accuracy on the test set. After incorporating the extensions discussed in the method sections, the sequential model achieved 65.6% accuracy. At a threshold T of 70% (i.e. only predict when $\Pr(y|\mathbf{x}) \geq T$), RNNs achieved a precision of 90% and recall rate of 59%. In practice, if a team is trying to scout their opponent and asked their scout to watch 3 games worth of video tapes, this algorithm could retrieve similarly many examples by watching about 5 games with only 10 percent erroneous examples. Again, a machine can watch as many games as deemed fit without requiring many resources. The speed for these algorithms at test time are all under a few second per 100 data entries where most time was spent in pre-processing the data.

3.2. Adapting to a new season (transfer learning)

Going into the next season, many strategies may remain the same (barring major coaching changes), but often the line-up changes. What remains is our interest to analyze them. We therefore test how well our algorithm works across seasons. In this section we investigate the first 3 months of the 2014-2015 data. To simulate the scenario of early in the season, the experiments are set up with a fraction of the data. We divide them equally into 3 batches chronologically, and we make only 1 batch available for training, 1 batch for validation, and test on the third batch, each roughly equivalent to a single month of the season.

For our model to do this, we can directly take a trained model, trained on the previous year’s data, and apply it to the new data. We call this “transfer”. On the opposite end of the extreme, we can train a new model using only the new data. We call this “new”. A hybrid, sensible approach is to start with the transfer model and then “fine-tune” it with the limited new data we have.

Using the same set of plays as trained in the previous season, there are 327 labelled strategies. Only 109 is made available for training for the ‘new’ and ‘fine-tune’ models. This set is slightly easier with a top-1 base-rate of 16.5%, and top-3 of 52.3%.

The ‘new’ model, due to the much smaller training set, only achieved 33.6% accuracy. The ‘transfer’ model performed slightly worse than it did in the old dataset. Most likely the plays have evolved slightly. Lastly, the ‘fine-tune’ model was able to adapt to the new dataset by just looking at the 109 examples, and achieve 61.9% top-1 accuracy.



Models	top-1 accuracy	top-3 accuracy	Precision/ Recall at T=.4	Precision/ Recall at T=.7
new	.336	.686	.336/.978	.347/.378
transfer	.537	.821	.541/1.0	.591/.958
fine-tune	.619	.873	.629/1.0	.639/.927

Table 3: Performance in the new season.

4. Conclusion

In this paper we studied the problem of offensive strategy classification in basketball. Many factors affect the result of this task. Some are intrinsic to the strategies themselves such as the complexity of interaction, distinctiveness, and diversity of the target classes. Other extrinsic factors such as reactions to defense, unexpected events such as fouls, and consistency of executions also complicate this task. However, the underlying dynamics, and rules of these strategies still allow human experts such as coaches to correctly label each sequence.

Using variants of neural networks, we demonstrated the possibility to automate play classification with promising results. With a top-1 classification accuracy of above 50%, simple NN gave us confidence that decent models can work well in this problem. With more understanding towards the data representation, and corresponding changes to a more advanced RNN model, we were able to achieve 66% top-1 accuracy, and 80% top-3 accuracy on unseen examples.

As an experiment to test how transferrable this method is across seasons, we used models developed on the 2013-2014 season to test in the 2014-2015 season. We show that with very limited data, these methods can leverage the representations learned from previous year, and still achieve reasonable performance.

Acknowledgment

The authors would like to acknowledge the support from Mitacs, and the Toronto Raptors in the form of a research grant, STATS LLC, and the NBA for the raw SportVU data. We would also like to thank the analysts from the Toronto Raptors, Keith Boyarsky, and Eric Khoury for unselfishly sharing their basketball knowledge and all of their constructive feedbacks.

References

- [1] J. Liu, J. Luo and M. Shah. Recognizing realistic actions from videos "in the wild", CVPR 2009, Miami, FL
- [2] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A Large Video Database for Human Motion Recognition. ICCV, 2011.
- [3] I. Goodfellow, A Courville and Y. Bengio. Deep Learning, Book in preparation for MIT Press 2015
- [4] S. Hochreiter, and J. Schmidhuber. "Long short-term memory."Neural computation 9.8 (1997): 1735-1780.