Building an On-Ball Screen Dataset Using Supervised and Unsupervised Learning

by

Neil Seward

A thesis submitted in partial fulfillment
of the requirements for the degree of

Masters of Applied Science

in

Electrical and Computer Engineering

University of Ontario Institute of Technology

Supervisor: Dr. Masoud Makrehchi

September 2018

Abstract

Applications of statistics and machine learning in sports analytics have made significant advances since the early days of Bill James' publication, Baseball Analyst. In the publication, analysts and researchers would investigate matters such as how often a batter would reach first base to determine which players were the most optimal to acquire. This type of data that recorded when players reached bases was relatively simple and cheap to acquire. Over time, more complex data became available to different professional sporting leagues.

In the National Basketball Association(NBA), cameras were installed in arenas to track player and ball movements. Movement tracking data enabled analysts and researchers to explore locations of players during in-game events, instead of providing insights from box score summaries and play-by-play data. The new tracking data has enabled teams to create insights from focus points that teams find valuable, such as on-ball screens. Important, detailed annotations like on-ball screens are not recorded in standard data sets. In order to capture the time stamps of the different events that are not recorded in standard data, teams have to employ analysts to record these time stamps and provide additional labels to the data.

As available performance data expanded, the scientific methods and tools used to analyze the data expanded as well. Recent advances in machine learning have shown different applications of neural network layers to be able to create abstract information from large and complex data sets. With the advance in applications of neural networks, several complex problems in sports analytics have been approached with this new technology.

We propose using unsupervised learning methods to create detailed labels of identified on-ball screen instances. To create the initial set of on-ball screen instances and to assist in proposing new instances, we use convolutional neural networks to classify positive and negative screens. Once our model is trained, we develop a framework to propose new annotation times to expand the data set, at the same time minimizing the amount of time for analysts to view game footage when verifying new screens. Using the established set of screens, we use several unsupervised learning methods to provide additional details to the screens identified. The new detailed labels provide additional insight into the screens identified as well establishing a framework to eliminate the costly process of creating multi-labelled instance of on-ball screens using human annotations.

Acknowledgements

I would first like to thank my supervisor, Dr. Masoud Makrechi. He has great patience and gave me the freedom I needed in my studies to approach topics, such as sports analytics, that may not be a common area of research in the lab. I would like to extend my gratitude to the Social Computing and Collective Intelligence Lab at UOIT for all of the help you have provided me, feedback given on my work, and the friendships I have developed.

I would also like to acknowledge Jackson Wang at the University of Toronto, who was a great mentor to me and helped guide me through a lot of the initial work that served as the basis for this thesis. Talking with him about different sports analytics papers was a real treat as it is rare to find researchers in this specific field. I would like to thank Keith Boyarsky as well for providing insight from an organizational standpoint on what professional sports teams look for in academic research.

Contents

List of Figures

List of Tables

Abbreviations

LDA Latent Dirichllet Allocation.

NBA National Basketball Association.

NMF Non-Negative Matrix Factorization.

PCA Principle Component Analysis.

ReLU Rectified Linear Unit.

RNN recurrent neural network.

seq2seq Sequence to Sequence.

Chapter 1

Introduction

1.1 Introduction to Basketball

In basketball, two teams play against each other over a period of four quarters, each quarter containing twelve minutes of action. At any time, the court contains five players from one team and five from the other team. In order for a team to win against their opponent, their offense must score as much as possible, as well as limiting their opponent's offense to less points than what they scored. In basketball, teams score when they successfully shoot the basketball into the net before the shot clock expires. Each basket scored can count for two or three points, depending on where the shot was taken on the court.

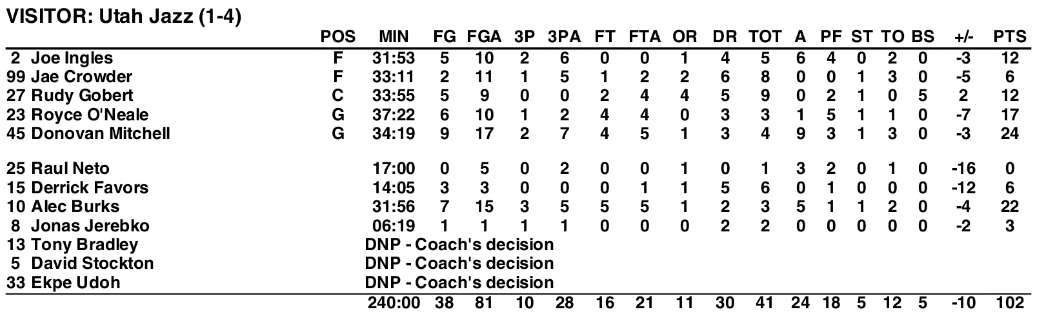
Basketball is a sport that provides continuous evaluation throughout a game, where almost all decisions in the sport can be evaluated to whether the decision contributed to a successful shot taken. Basketball's evaluation method is very unique compared to other sports. In hockey, teams do not score as often as in basketball, sometimes not even scoring once in a single game. Without a high scoring game, it becomes relatively difficult to attribute players' actions to a valuable metric. Although baseball teams do not score as frequently as basketball, baseball is the only other team based sport that discrete evaluations are valuable. Player actions in baseball always result in positive and negative actions, where each action can be attributed to a ball, strike, out, taking base, or scoring a run. For analysts in baseball and basketball, overall game summaries (also known as box-scores) and game logs (also known as play-by-play) have been the main source of data for research now for decades. For basketball, these summaries would contain numerous descriptions of common events in the game, such as baskets scored, assists, etc.

1.1.1 Basketball Glossary

- **Shot:** offensive scoring attempt when a player forces the ball to leave his hands and towards that basket
- **Pass:** offensive player in possession of the ball throws the ball to another player on offense
- **Possession:** a 24 second period of time where one team is tasked with scoring and the other team is tasked with defending
- **Play:** a team on offense makes a series of consecutive moves and passes among players in order to get an effective shot during a possession
- **Guard:** player on defense designates a player to prevent scoring or any other success on the offensive end by preventing passes, shot, etc.
- **Screen:** an offensive tactic used to free an offensive player of the defensive player that is guarding them

1.2 Motivation

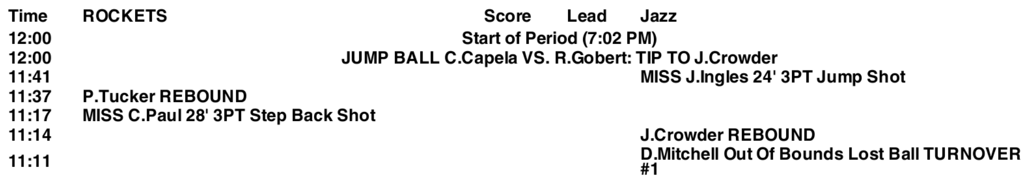
Although detailed, game summaries in basketball are limited to conventional events. These box scores, seen in Figure 1.1, and play-by-play summaries, as portrayed in Figure 1.2, do not contain information on offensive or defensive play information. For some teams, these specific annotations are extremely valuable when evaluating player performance and observing play execution. In order for teams to get annotation information beyond the standard game summaries and play-by-play information, they employ analysts to create human labels. For this task, analysts would spend an entire game with their notebook, writing down when a specific action occurs and the players specifically involved in the action. Because there is so much to record in such a short period of time, analysts can miss actions, or make mistakes when identifying players involved in actions.



VISITOR: Utah Jazz (1-4)

	POS	MIN	FG	FGA	3P	3PA	FT	FTA	OR	DR	TOT	A	PF	ST	TO	BS	+/-	PTS
2 Joe Ingles	F	31:53	5	10	2	6	0	0	1	4	5	6	4	0	2	0	-3	12
99 Jae Crowder	F	33:11	2	11	1	5	1	2	2	6	8	0	0	1	3	0	-5	6
27 Rudy Gobert	C	33:55	5	9	0	0	2	4	4	5	9	0	2	1	0	5	2	12
23 Royce O'Neale	G	37:22	6	10	1	2	4	4	0	3	3	1	5	1	1	0	-7	17
45 Donovan Mitchell	G	34:19	9	17	2	7	4	5	1	3	4	9	3	1	3	0	-3	24
25 Raul Neto		17:00	0	5	0	2	0	0	1	0	1	3	2	0	1	0	-16	0
15 Derrick Favors		14:05	3	3	0	0	0	1	1	5	6	0	1	0	0	0	-12	6
10 Alec Burks		31:56	7	15	3	5	5	5	1	2	3	5	1	1	2	0	-4	22
8 Jonas Jerebko		06:19	1	1	1	1	0	0	0	2	2	0	0	0	0	0	-2	3
13 Tony Bradley		DNP - Coach's decision																
5 David Stockton		DNP - Coach's decision																
33 Ekpe Udoh		DNP - Coach's decision																
		240:00	38	81	10	28	16	21	11	30	41	24	18	5	12	5	-10	102

Figure 1.1: Box Score Sample [1]



Time	ROCKETS	Score	Lead	Jazz
12:00		Start of Period (7:02 PM)		
12:00		JUMP BALL C.Capela VS. R.Gobert: TIP TO J.Crowder		
11:41				MISS J.Ingles 24' 3PT Jump Shot
11:37	P.Tucker REBOUND			
11:17	MISS C.Paul 28' 3PT Step Back Shot			
11:14				J.Crowder REBOUND
11:11				D.Mitchell Out Of Bounds Lost Ball TURNOVER #1

Figure 1.2: Play By Play Descriptions [1]

1.2.1 Player Tracking Data

Our work utilizes the publicly available SportVU tracking data from the 2015-16 NBA season [28], [27]. This tracking data contains the X,Y positions of ten players, and the X,Y,Z positions of the ball. The data is recorded for the entire game at 25 frames/second. An example of the player tracking data can be seen in Figure 1.3. To visualize the movement of the player over time, a fading effect from light to dark is added to represent the start time to end time.

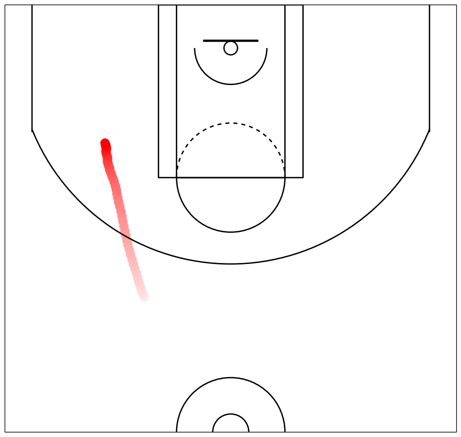


Figure 1.3: SportVU Movement Data

1.3 Problem Description

Regarding the research presented, the specific action we want to provide additional context for is the on-ball screen. On-ball screen actions are a specific offensive strategy in basketball that involves using a stationary player offensive player that is not in possession of the ball as a wedge to free the ball-handling player of the defender guarding them. In the on-ball screen actions, there are two main acts: the approach of the players to the screen, seen in Figure 1.5a, and the execution after the screen was utilized, seen in Figure 1.5b. There are instances of screens where the screen

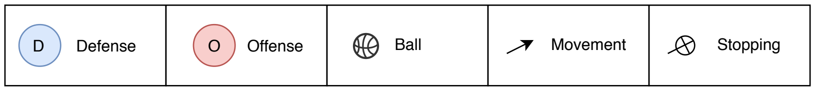
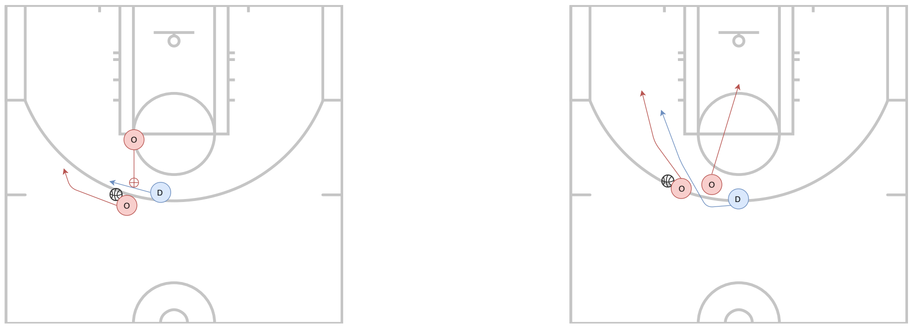


Figure 1.4: Movement Plot Legend



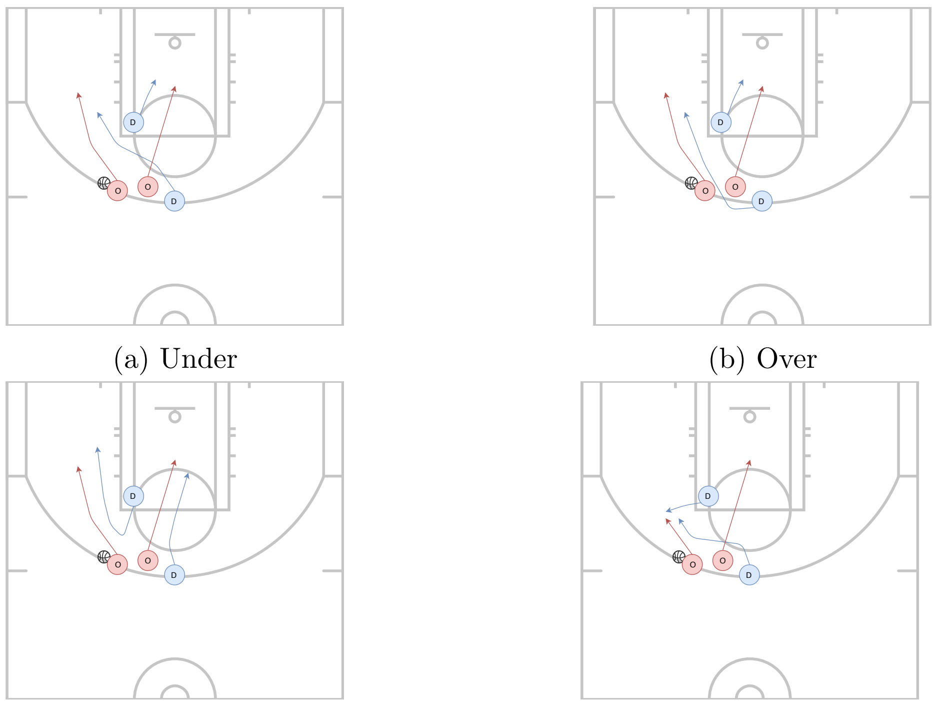
(a) Approach (b) Execution

Figure 1.5: On-Ball Screen Approach and Execution

is set for players without possession of the ball, otherwise known as off-ball. Our research specifically is focused on identifying on-ball screens and not off-ball screens.

In Figure 1.4, the ball handler belongs to the offense, O, and is in possession of the ball. The player defending the ball handler belongs to the defense, D. The player that sets the screen to impede the on-ball defender is part of the offense. For this specific instance of an on-ball screen, the on-ball defender follows the ball-handler through the screen, but travels under the player setting the screen instead of moving over the screen. Because the on-ball defender moves under the screen, the specific defensive label for this instance would be an under.

Not every action sequence for a screen is executed on defense in the same fashion. There is no defined number of different types of on-ball screens in basketball, but one can define a set of screen outcomes for classification purposes. In Wiens et al., they define four distinct types of on-ball screens for defensive execution schemes: the on-ball defender goes *under* the screen, the on-ball defender goes *over* the screen, the on-ball defender *switches* with the screen defender, and the on-ball defender and the



(a) Under (b) Over

(c) Switch (d) Trap

Figure 1.6: On-Ball Screen Defensive Schemes

screen defender *trap* the ball handler [31]. The four defensive execution schemes can bee seen in Figure 1.6.

1.4 Proposed Solution

One of the initial goals of this research was to create a standard data set for on ball screens utilizing available movement data. By outlining these specific actions, we can use these on-ball screen actions to continue this research to provide additional labels for defensive execution schemes. By starting with an initial ground truth set, we are able to develop a classifier using convolutional networks to observe raw games outside of the training set to detect new screen actions in unseen games. Using detection on the raw probability outputs from unseen games, we can propose screen annotations, minimizing the time needed to label screen annotations from new games.

Our work focuses on finding the true labels behind on-ball screen actions. We provide a methodology for analysts to create a set of defined on-ball screen actions. Our research showcases an unsupervised learning application to create additional labels for identified on-ball screen annotations. In this work, we show that behavioural and domain specific feature representation can act as input in sequential learning methods. The encoded state that is learned from feature representations of trajectory sequences can be used as a vector representation of a player's action during an on-ball screen. We represent these actions as words in an on-ball screen vocabulary and create relative text documents for a screen. Once we create a data set of text documents and a corresponding vocabulary for on-ball screens, we use topic modelling techniques to identify additional labels that we can provide to identified on-ball screen actions.

1.5 Contributions

- In order to create the framework for proposing new screen annotations, we train convolutional neural networks on a ground truth set of human annotations. Although this is not the first application of machine learning models for the task of binary classification of on-ball screens, the research presented the first known use of deep learning models when classifying these actions. The trained classifier obtains a validation accuracy of 91%.

- On top of building a classifier, we describe our approach to proposing new annotations in unseen games. Using non-max suppression to smooth raw output from the trained model, our framework creates precise predictions of when on-ball screens occur in games outside the ground truth set. When detection new annotations, our approach obtains a recall of 90% and a precision of 40%. Because our approach creates input without any prepossessing to identify domain specific features, the annotation proposals can be made in a much shorter amount of time.

- Once we have created a base set of on-ball screen annotations, our framework enables analysts to create additional labels for the four defensive schemes provided in Figure 1.6. We apply unsupervised learning methods to enable basketball teams to create detailed on-ball screen annotations using standard SportVU tracking data.

- Using a set of identified on-ball screens, our framework draws from different topic modelling techniques used in text processing to identify "topics" or specific types of executions on both offense and defense for on-ball screens. In our research, we represent an on-ball screen as a text document. The trajectories of each of the players involved in the screen annotations are represented within a vocabulary set of actions. We use unsupervised learning techniques similar to Neural Machine Translation learning to identify actions in player trajectories. As input to the model that identifies actions from player trajectories, we use a combination of behavioural and domain specific features in a sequence to describe a player's movement.

- Once a vocabulary of screen actions is set for the approach and the execution, we combine actions at each phase of the screen to form pairwise "words" to use for topic modelling. Using the contextual vocabulary, we form text documents of each screen annotation as input to a Latent Dirichlet Allocation(LDA) model. As output from the LDA model, each of the screen annotations is assigned to one of four topics, representing the defensive execution scheme of the screen. Using LDA to provide class outputs, we achieve an average precision of 37% and an average recall of 35% on all four classes.

1.6 Organization of Thesis

The thesis proceeds as follows. In Chapter 2, background is presented to the different neural network architectures used. In Chapter 3 overviews of related research in machine learning applications in sports analytics are given. In Chapter 4, we provide the framework used to create an on-ball screen classifier and detection system using convolutional neural networks. In Chapter 5, we further explore the ground truth annotations using sequence to sequence models and topic modelling, and create additional labels that depict the defensive execution scheme of the screens. In Chapter 6, the research presented in this thesis is concluded and future works are outlined.

Chapter 2

Background

2.1 Neural Network Architectures

Specifically in this thesis, the models used at the core of the work contributed involve different variants of neural networks. Neural networks has seen success in predicting simple game outcomes and tournament champions in sports analytics research [17], [6], [30]. Applications of complex neural networks using movement tracking data is relatively infant, where the bulk of the research using this data has been done since 2016 [39], [18].

2.1.1 Feed Forward Neural Networks

Neural networks are systems of neurons and neuron connections that are tuned through gradient descent optimization as machine learning models [10]. The operations of these networks can be explained as functions of a brain, where signals coming from a neuron in the brain depend on previous signals from other neurons as input. In neural networks, the neurons are different than the human brain.

In neural networks, input to neurons are a product of the previous neurons' out-

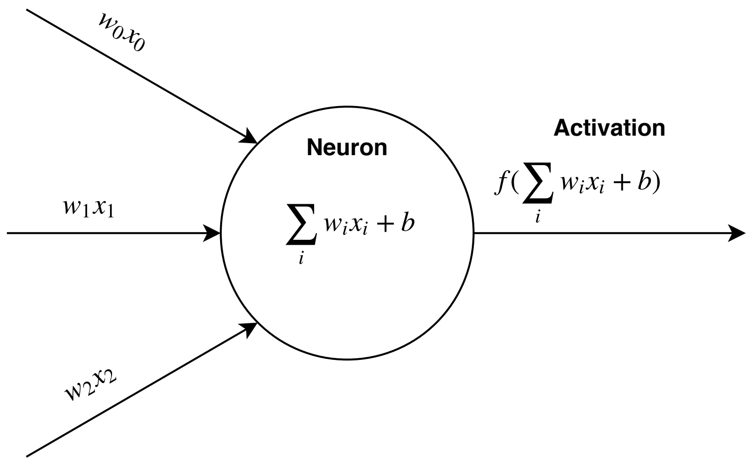


Figure 2.1: Sample Neuron

puts and the corresponding weights for these neurons. The neuron produces a single output as a product of an activation function on the neuron inputs. A sample neuron, along with the weighted input and activated output, can be observed in Figure 2.1. In the different architectures applied throughout the research, the activation that is commonly used in different architectures is the Rectified Linear Unit activation function(ReLU) [12]. Depending on the network, we also apply a dropout regularization method to prevent over fitting [15].

In feed forward networks, the overall structure beyond the internal of the neuron is relatively simple. These networks have a raw input layer and a single output layer for the prediction task at hand. In between the input and output layers in the network are hidden layers. These hidden layers can be of higher dimensional than the input layers, but can also compress the input by using less neurons than the input layer, depending on the purpose of the prediction task [20], [13]. A representation of a three-layer feed forward network can be seen in Figure 2.2. The output layers produce inputs for gradient descent optimization, which is used to tune the weights of the hidden layers through backpropagation [10].

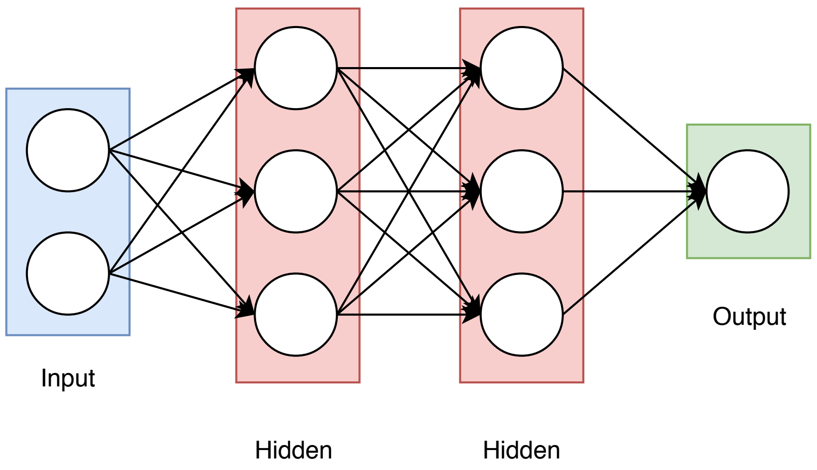


Figure 2.2: Feed Forward Neural Network

2.1.2 Convolutional Neural Networks

Convolutional networks are an advanced form of neural networks that perform especially well with images. The components in a convolutional network layer are similar to a hidden feed forward layer. Both contain contain weights and biases that are tuned, but where feed forward layers are comprised of a single row of neurons, convolutional layers have three dimensions of neurons [22]. Because of the shape of the convolutional layer, these networks have the capability to handle much larger data inputs, such as image data. A sample architecture of a convolutional neural network is portrayed in Figure 2.3.

An additional feature of a convolutional layer is the operations of filtering and pooling. Convolution filters produce different levels of abstraction from the input as a product of kernel-sized filter matrix multiplication. As the network become deeper with more layers of convolution filters, higher levels abstraction can be extracted. Max-pooling layers summarize subsections of the outputs produced by filters, by selecting the maximum output from matrix multiplication, subject to kernel size and stride size [22].

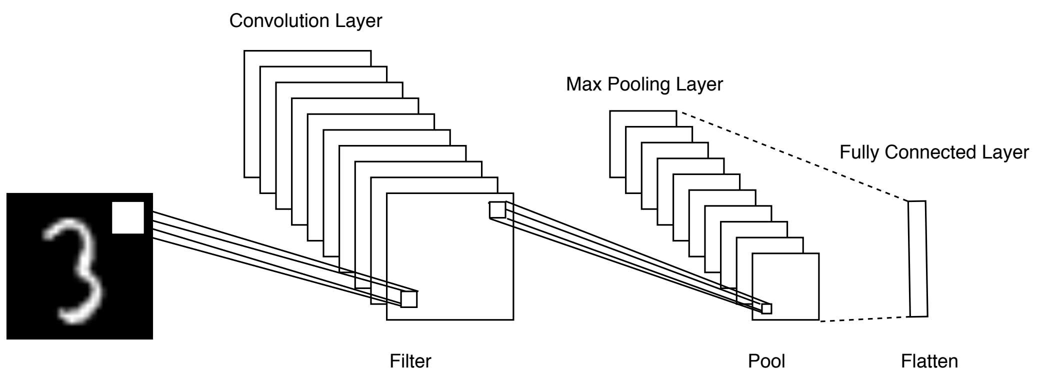


Figure 2.3: Convolutional Network Architecture

2.1.3 Sequence to Sequence Learning

Sequence to Sequence(seq2seq) learning is an applied form of deep learning that uses recurrent neural network(RNN) layers to process sequential data. A key building block in seq2seq models is the recurrent layer. Recurrent neural networks are neural networks where the neurons in the layers have the ability to "remember" hidden information and utilize memory for future predictions [9]. The use of hidden units in RNN prediction can be defined as

$$h_t = f(W_t x_t + U_{t-1} h_{t-1} + b_t) \tag{2.1}$$

Where the hidden (h_t) unit at time t consists of a non-linear activation function f, such as ReLU [12] that receives input from x_t and the previous hidden unit h_{t-1}. The previous hidden unit at time $t-1$ is amplified by the hidden weights U and the input at time t is amplified by the weight matrix W. An illustration of a recurrent layer is depicted in Figure 2.4.

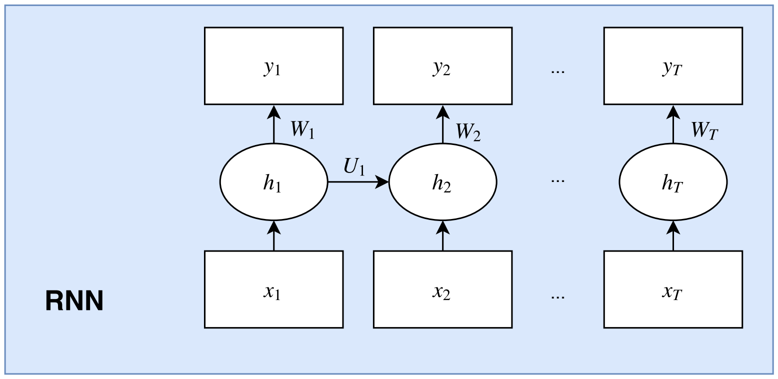


Figure 2.4: Recurrent Neural Network Architecture

In the past, seq2seq learning has been used in language translation tasks where an input of a sentence from one language is given and the model is trained to produce the same sentence, but translated to a target language [2], [38]. The sequence input is fed into an encoder network, consisting of a RNN layer. As the input is fed through, the encoder provides a single encoded state. The encoded state serves as input, along with additional decoding input to a decoder network. The decoder network consists of an additional recurrent network layer that produces a target sequence from the encoded state and the decoder input. In Figure 2.5, the encoder receives an input $\{x_1, x_2, ...x_t\}$ and produces an encoded state z. The decoder receives the encoded state z as the initial hidden state as well as separate input $\{y_1, y_2, ...y_{t-1}\}$, which produces an output sequence $\{y_1, y_2, ...y_t\}$. After the decoded output is produced, it can be compared to the target sequence to calculate the reconstruction loss. For sequence to sequence learning, the goal is always to produce a decoded sequence that is as close to the target sequence as possible by minimizing the loss from reconstruction. To do this, the reconstruction loss can be defined as

$$Loss = \sum_{t=1}^{n} \|y_t - \hat{y}_t\|_2 \tag{2.2}$$

Where y_t is the decoded output at time t, $\hat{y}_t$ is the target output at time t. A depiction of the encoder and decoder network layers in a seq2seq architecture can be observed in Figure 2.5.

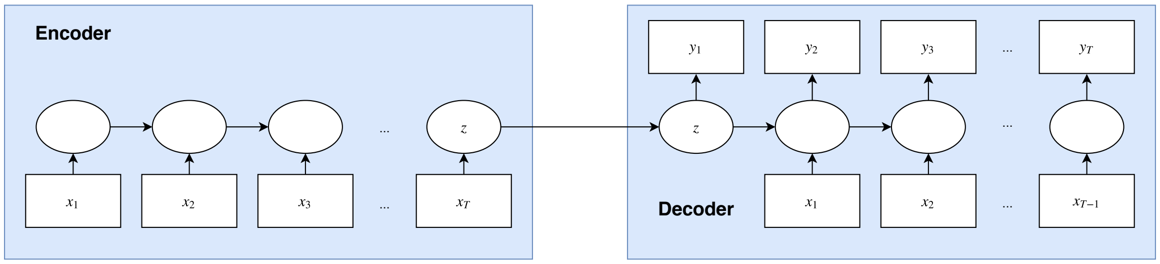


Figure 2.5: Seq2Seq Architecture

2.2 Topic Modelling

Topic modelling is an unsupervised learning method most commonly associated with text processing. In text processing, topic modelling is applied to find hidden meanings behind bodies of text by placing documents into categories, or ”topics”. These documents are assigned into topics entirely based on their underlying content; how that content relates to all possible topics, and how each item in the document's content relates to the topics.

For understanding, we define the following terms [5]:

- A *word* is the one-hot encoded representation of an element in a vocabulary set, denoted by $\{1, ..., V\}$, where the V^{th} word in the vocabulary is denoted by $w^v = 1$ and $w^u = 0$ for all $u \neq v$.
- A *document* is a collection of N words denoted by $w = \{w_1, ..., w_N\}$.
- A *corpus* is a collection of M documents denoted by $C = \{d_1, ..., d_M\}$.

2.2.1 Latent Dirichlet Allocation

Latent Dirichllet Allocation(LDA), is a latent variable model, similar to Non-Negative Matrix Factorization(NMF) [25], and Principle Component Analysis(PCA) [8]. LDA can be applied to a corpus of sports articles to find words related to different sports like basketball(”dunk”, ”alley-oop”, ”three-point”), hockey(”icing”, ”puck”), volleyball(”spike”, ”serve”), etc.

In LDA, each document has a probability distribution for k different topics, θ_d [5]. From the document d, each word in the document has its own probability distribution inferred for topics $\{1, ..., T\}$, z_{d_n}. From the topic distribution z_{d_n}, words can be inferred. A depiction of the topic distributions for a text document can be seen in Figure 2.6, where the input to the LDA model is the Bag-of-Words representation of the vocabulary within the document.

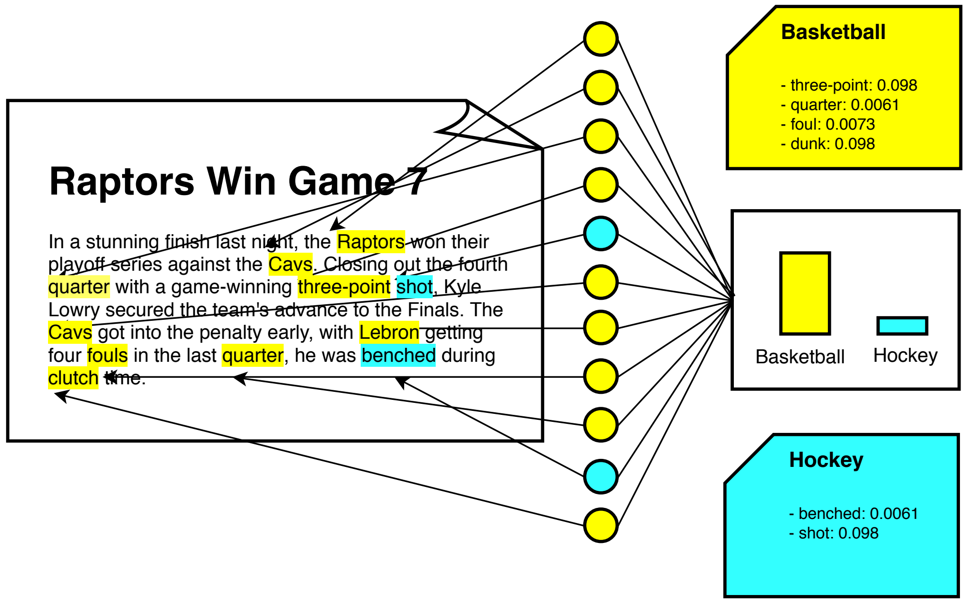

Raptors Win Game 7
In a stunning finish last night, the Raptors won their playoff series against the Cavs. Closing out the fourth quarter with a game-winning three-point shot, Kyle Lowry secured the team's advance to the Finals. The Cavs got into the penalty early, with Lebron getting four fouls in the last quarter, he was benched during clutch time.
Basketball
- three-point: 0.098
- quarter: 0.0061
- foul: 0.0073
- dunk: 0.098
Basketball
Hockey
Hockey
- benched: 0.0061
- shot: 0.098

Figure 2.6: Topic Distribution over Document

2.3 Embedding and Clustering Methods

2.3.1 T-Distributed Stochastic Neighbour Embedding

Embedding is the process of representing high dimensional data in a low dimensional space. This practise is often done to reduce high dimensional data, like images, into vector sizes that can be easily used in clustering for further interpretation. A popular form of embedding in machine learning research is T-Distributed Stochastic Neighbour Embedding(t-SNE), [14]. In the high dimsensional representation, t-SNE models similarity between points as join probability distributions. When t-SNE is applied to high dimensional data, the model minimizes the KL-Divergence in the probability distributions of the high dimensional data and the low dimension embeddings. As the model is optimized, similar points in high dimension space are pushed closer together in the embedded space, while dissimilar points are pushed farther apart in the embedded space. [23].

Because t-SNE does not maintain distances between points on separate embedding runs [14], many have cautioned against using the embedded results as input for clustering algorithms. Although t-SNE is known not to be reliable for clustering, it has shown promise when embedding SportVU movement data and other features that can be used to assign clusters in sports data for further analysis, including in trajectory analysis [40], [33].

2.3.2 K-Means Clustering

K-Means is a clustering analysis tool that aims to assign n data points to a predetermined k clusters [29]. Each of the points is assigned to a cluster based on how close the point is to the means each of the clusters. The cluster which has the smallest distance to its mean is assigned to its population. Once all of the data points are assigned to clusters, the means of each cluster population is recalculated. With the new means, the data points are re-assigned based on the new distances to each mean. The entire process of assigning to cluster populations and re-calculating means is repeated until there is convergence among the assignment of points to cluster centers.

Chapter 3

Related Work

3.1 Predicting Shots Using Spatiotemporal Features

One of the most researched areas in sports analytics has been predicting shot success. Predicting whether a scoring attempt is successful is key for team management and analysts when evaluating player and team performance. With a strong prediction model, teams have the ability to determine the value behind decisions made by players before attempts are made [7].

There has been several different models produced that approach the problem of shot prediction using movement tracking data. In the realm of soccer, Lucey et al. found that including descriptions of the offensive play(free-kick, counter-attack) and other contextual information produced a robust logistic regression model [7]. For basketball, Harmon et al. found that channel information can be used to portray player information in an image representation of a possession as input to a convolution network prediction model [11].

3.2 Predicting Trajectories Using Recurrent Networks

With the introduction of complex deep learning frameworks came more complex problems in sports analytics. Predicting trajectories in sports has been shown to have many different applications, including determining optimal defense positioning [24]. Using a combination of recurrent network layers, Zheng et al. were able to create a model that predicts player trajectories in basketball through a combination of macro and micro goals [42]. Using recurrent models presented originally in handwriting generation, Shah et al. used predictions of the basketball trajectory to predict whether a shot was successful [37].

3.3 Identifying On-Ball Screen Annotations

Previously [32], [18], there has been work to show how accurate machine learning models can be used to propose event annotations. In the work presented by Wiens et al. [32], annotation times are proposed using a set of logic rules surrounding stationary ball and player movement. Not all of the proposals are actually on-ball screens, but contain movement that satisfies the logic. Once the proposals are made, human annotations are created from the proposals that creates positive and negative labels from the proposals. For the classification task of whether a proposal is positive or negative, features are extracted. These features involve pairwise distances that involve the ball, as well four as roles identified for players involved in the screen action.

Once the proposals are made, summary pairwise distance features are extracted for the screen roles for both their approach and execution after the screen was set.

Each feature is binarized into five bins. Using a support vector machine model, they are able to identify both positive and negative screens.

3.4 Classifying Player Actions

One of the first applications of neural networks using tracking data has been identifying plays and offensive strategy, [39], [35], [36]. In work presented by Wang and Zemel, they were given a set of offensive plays to classify using convolutional networks [39]. As part of the input to the network, a pictorial representation was given utilizing different coloured channels to represent different roles between the offense, defense, and the ball. The roles represented in the image went far beyond just three categories of offense, defense, and ball and were formed to represent player position on offense (point guard, shooting guard, etc). The roles were formed from an auto-encoder neural network that found the "true" position details of each offensive player, placing them in one of five roles.

3.5 Trajectory Cluster Identification

In the realm of spatial-temporal data mining, trajectory clustering is a key tool used when identifying trajectories like shipping routes [41], and player trajectories [33], [16]. There is a need to identify paths taken by objects over a period of time. By identifying these trajectories through unsupervised methods, we can understand the underlying movement patterns among a large set of trajectories.

In the previous clustering works, trajectories were compared through different distance similarity measures [3]. These methods were useful for finding similar trajectories in a confined space and over a similar period of time, but do not have the ability to find similar clusters regardless of location and time variance. In work presented by [41], trajectories can be clustered using behavioural representations and deep learning. Because the trajectories can be represented as a sequence of behavioural features, the deep learning methods used were able to find similar clusters, regardless of the longitude and latitude positions.

3.6 Topic Modelling in Basketball

Luke Borne and Alex Miller [33] employ unsupervised learning techniques to create labelled offensive possessions using topic modelling practises. To create a vocabulary of words from the movement in a possession, player actions are first extracted by segmenting movement by acceleration logic. Once these actions are extracted, the underlying movement in each action is represented by Bezier curve parameters that can be used in clustering algorithms. Each of the clusters formed from the actions can be considered a single word in a vocabulary of player actions.

Once the actions are extracted and the vocabulary is formed, the possession can be represented as a text document of actions identified. The model used to identify topics from the text documents formed is Latent Dirichlet Allocation [4].

Chapter 4

On-Ball Screen Classification and Proposals Using Convolutional Neural Networks

4.1 Obtaining On-Ball Screen Annotations

Because our later work aims to provide detailed labels beyond simple on-ball screen identification, our unsupervised model requires a large set of on-ball screens action sequences. Inspecting 14 games in the 2015-16 NBA season, we identified approximately 1200 on-ball screen instances. For future use in identifying on-ball screen instances, we trained a convolutional neural network to use in proposing future on-ball screen instances in order to minimize watching game footage when creating human annotations. We approach creating new annotations using detection algorithms on raw probability outputs throughout possessions.

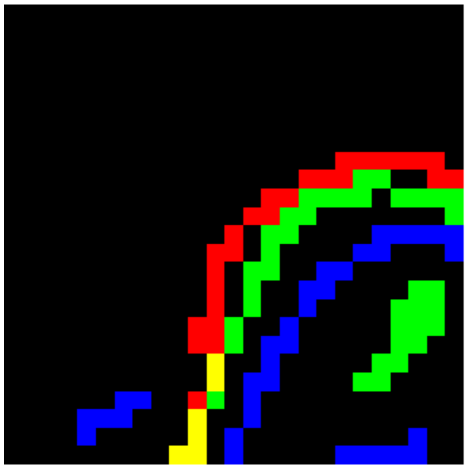


Figure 4.1: Example On-Ball Screen Action Inputs

4.2 Data

As input into our image, we construct snapshots that contain movement trajectories of players on offense and defense, as well as the ball. The extractor produces an image over a specified window with trajectory channels for offense, defense, and the ball. In the images, the tracked movement of players on offense and defense is limited to a 25x25 pixel window surrounded by the ball as the window's center. The ball is red, players on offense are green, and players on defense are blue. To see an example input into the network, it is shown in Figure 4.1. For each instance, we define a four second window to provide the entire context of the approach and the execution of the screen is set. We assume that the entire approach of the screen starts within the two seconds before the screen time and we assume that the execution scheme of the screen is finished in the two seconds after the screen time.

4.3 Model

Our classification model consists of three convolution layers, each with max-pooling layers, seen in Figure 4.2. The model's learning rate begins at 0.001 and decays

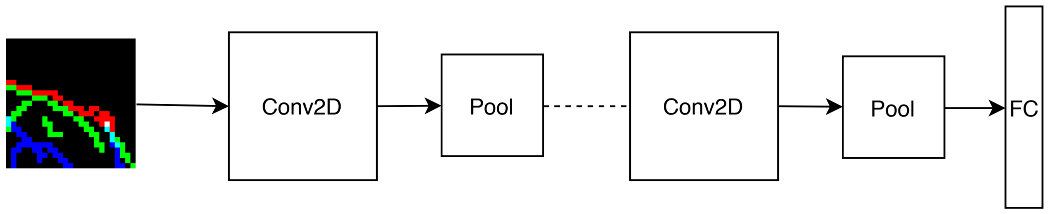


Figure 4.2: Architecture of Convolutional Network

at a rate of 0.1/3000 iterations over 10 000 iterations. Through trial and error, it was determined that a higher learning rate of 0.01 was too high to find the optimal trained model, and a smaller learning rate of 0.0001 was too slow to achieve the best trained model in a timely fashion. Although we started with a learning rate of 0.001, the learning rate needed to shrink as the model changed as the loss values became smaller with the optimization of the model's weights. We applied dropout layers to each pooling layer and the fully connected layers with a dropout rate of 0.2. To prevent over fitting when training the model, we introduced dropout. We train the model on a batch size of 64. In order to train the network on such a large set of data, we decided to use batch processing.

We optimize our model by minimizing a combined loss that is dependant on cross entropy and L2 regularization [34], with a standard Adam optimizer [19]. The representation of the combined loss can be seen below.

$$Loss = -\sum_i y_i \log \hat{y}_i + \frac{1}{2}\lambda w^2 \tag{4.1}$$

Where y_i is the ground truth of class i and $\hat{y}_i$ is the raw probability output from the model for class i, representing the cross entropy loss term. The L2 regularization term penalizes the loss as a product of the regularization constant λ and the square of the weights w.

4.4 Classification

Once finished with the human annotations, our training data set contains 12 annotated games with 1028 annotations, and a validation set with one game and 92 annotations. The model is trained on a 50/50 split between on-ball screen action sequences and non screen action sequences. For validation, the model is evaluated on a separate 50/50 split.

In Table 4.3, we compare our method of using a convolutional network to the methods previously presented by Wiens et al., where they train a support vector machine model on mutli-agent pairwise distance features [32]. These features are described in Table 4.1 and are formed during the approach and execution phases of the screen.

Each of the features from both the approach and the execution are binned into quintiles. We describe these specific features in detail for context when comparing our model to the baseline model described by Wiens et al. [32]. In order to prepare these features as input, we need to create a method to identify the four player roles involved from the tracking data. We develop the framework to identify these roles in Section 4.4.1. For each player in the four identified roles, the pairwise features listed in Table 4.1 are calculated against the ball, and the three other role players.

Feature	**Equation**
Minimum distance between player a and player b at the moment of the screen	$\min_t d_{t,a,b}$
Sum of distance between player a and player b over the approach window	$\frac{1}{s}\sum_{t=1}^{s} d_{t,a,b}$
Difference in distance between player a and player b at the start of the screen window and the moment of the screen (approach)	$\frac{(d_{s,a,b}-d_{1,a,b})}{s}$
Difference in distance between player a and player b at the moment of the screen and the end of the screen window (execution)	$\frac{(d_{n,a,b}-d_{s,a,b})}{n-s}$
Sum of distance between player a and player b over the execution window	$\frac{1}{n-s}\sum_{t=s}^{n} d_{t,a,b}$

Table 4.1: Wiens Features for Screens

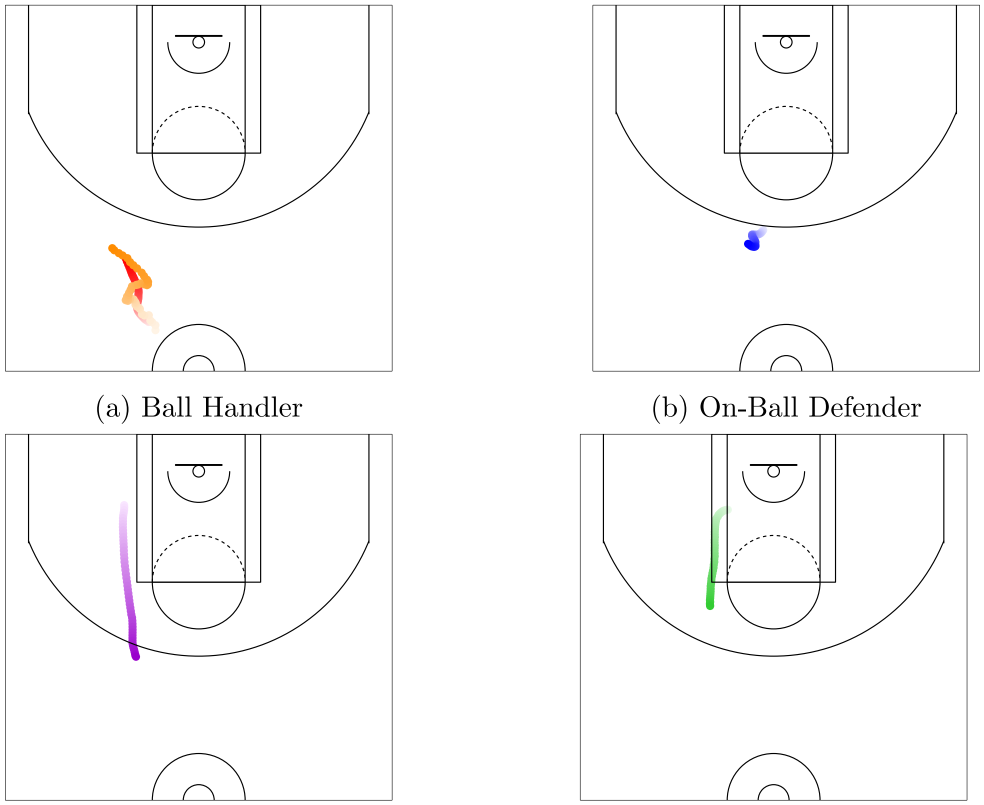
4.4.1 Role Identification

In all on-ball screens, there are traditionally four players involved in the action. For the purposes of the thesis, we assume that there are always the same four roles in each on-ball screen. Each of these four players have unique offensive and defensive roles during a screen action. In order to extract these roles from the action sequence, we observe movements by players during the approach phase of the screen. We determined different hard logic rules for the four different roles in the action: ball handler, on-ball defender, screen setter, and the screen defender.

The logic we present determines the four roles for the screen annotation using only the approach window and excludes the execution window. For these equations, we define the approach window from the beginning of the screen, where time $t = 1$ to the screen moment $t = n$. The ball handler(bh) is the player on the offensive team(Off) with the smallest mean squared error with the ball(b). We determine the on-ball defender(bd) to be the player on defense(Def) that has the smallest mean squared error with the ball handler. The screen setter(ss) is the player on offense, aside from the ball handler, that is closest to the ball at the moment of the screen, n. The screen defender can be defined as the player on defense, aside from the on-ball defender, that is the player with the smallest mean squared error with the screen setter. These extracted roles can be observed in the approach windows shown in Figure 4.3. We provide equations for the role formation in Table 4.2.

Role	**Equation**
Ball Handler(bh)	$bh = \min_{(p \epsilon Off)}\{\frac{1}{n}\sum_{t=1}^{n}(b_t - p_{it})^2\}$
Ball Defender(bd)	$bd = \min_{(p \epsilon Def)}\{\frac{1}{n}\sum_{t=1}^{n}(bh_t - p_{it})^2\}$
Screen Setter(ss)	$ss = \min_{(t=n \mid p \epsilon Off \mid p_i \neq bh)}\{\lvert\lvert b_t - p_{it}\rvert\rvert\}$
Screen Defender(sd)	$sd = \min_{(p \epsilon Def \mid p_i \neq bd)}\{\frac{1}{n}\sum_{t=1}^{n}(ss_t - p_{it})^2\}$

Table 4.2: Screen Role Determination



(a) Ball Handler

(b) On-Ball Defender

(c) Screen Setter

(d) Screen Defender

Figure 4.3: Roles Extracted From Approach

Model	**Accuracy**	**Loss**
Convolution Network	0.91	0.27
Wiens Features + Support Vector Machine	0.91	0.18
Wiens Features + Logistic Regression	0.91	0.24
Wiens Features + Random Forest	0.88	0.29
Wiens Features + Naives Bayes	0.91	0.65
Wiens Features + Feed Forward Network	0.9	0.29
Wiens Features + Gradient Boosting	0.82	0.36

Table 4.3: On-Ball Screen Classification Results

From observing the classification results, our convolution network approach with raw image input performs as well as previous methods used for screen classification. Without the pre-processing needed for the Wiens Features described in Table 4.1, we can create a model for classification that performs as well as models that require costly features.

4.5 Detection

With the goal in mind of developing a model that can best detect new screen actions, we evaluate several models on how they perform detecting these actions on new games. To ensure consistency, we separated one game from the training data set for testing purposes. When evaluating the performance of the different models, we applied a Non Max Suppression detection algorithm that uses the raw probability outputs from unseen annotations.

4.5.1 Non Max Suppression

In Non Max Suppression, we applied a smoothing function over a specific time window throughout a probability map. For the smoothing function, we applied the max probability of a specified window to the entire window. Once the entire instance is smoothed, all probability windows that do not meet a certain threshold are suppressed. The proposed centers of the screen actions are the non-suppressed local maximum at each window radius. The screen proposal algorithm is detailed in Algorithm 4.1 and the non-max suppression algorithm used for screen proposals is detailed in Algorithm 4.2.

When evaluating the results of proposing screen annotations, we use precision and recall. Let P represent Precision, where TP are the true positives identified and the

Algorithm 4.1 Screen Proposal Algorithm

```
1: procedure PROPOSE($event, k, radius, model, threshold$)
2:     $event.moments \leftarrow \frac{event.moments}{k}$          ▷ Down sampled by factor of k
for each: moment in event.moments
3:     $annotation \leftarrow make\_annotation(moment, radius)$
4:     $features \leftarrow get\_features(moment, annotation)$
5:     $probs \leftarrow predict\_proba(features, model)$
6:     $proposals \leftarrow NMS(annotation, probs, threshold, radius)$
7:     return $proposals$
```

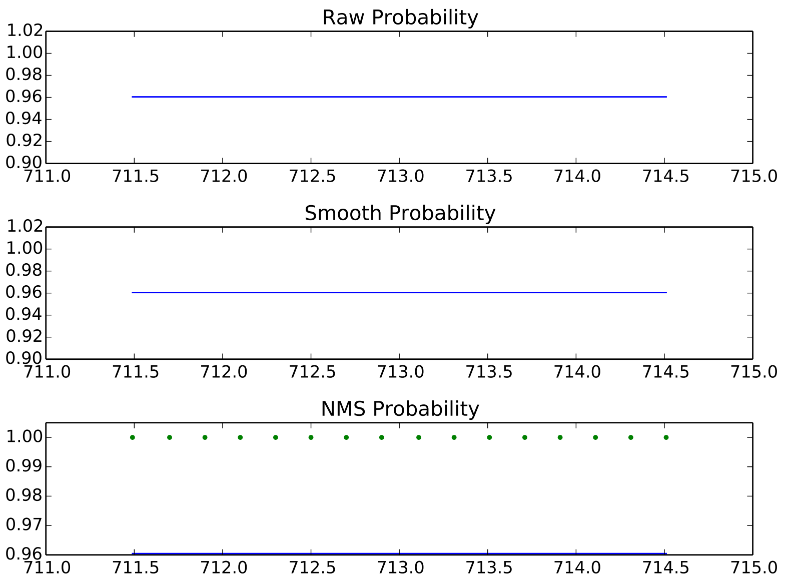
Algorithm 4.2 Non-Max Suppression Detection Algorithm

```
1: procedure NMS($annotation, probs, threshold, radius$)
2:     $probs \leftarrow smooth(probs)$
3:     $probs[probs < threshold] \leftarrow -1$
4:     $proposal\_centers \leftarrow arg\_local\_maximas(probs)$
for each: proposal_center in proposal_centers
5:     $proposal \leftarrow make\_window(proposal\_center, radius)$
6:     return $proposals$
```

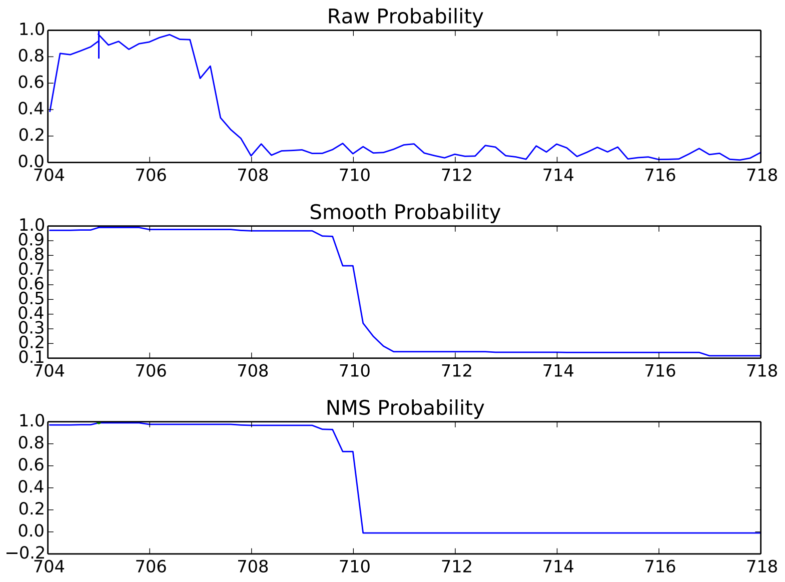
FP are the false positives identified. Precision can be defined as $P = \frac{TP}{FP+TP}$. Let R represent Recall, where FN represent the false negatives classified. Recall can be defined as $R = \frac{TP}{TP+FN}$.

The detection process is focused on identifying true positives throughout an entire game. There can be an infinite number of false action sequences that are identified throughout that game, but only n possible sequences can be properly detected as true positives in the game data. The best way to evaluate a model's ability to detect these n screens in a new game is to see how true the proposals are through precision, and to evaluate the model's ability to identify all n screens through recall. It is not practical to evaluate accuracy in this setting.

The effect of non-max suppression on raw probability outputs can be seen in Figure 4.4. In the last plot in Figure 4.4, the green dots are the proposal centers referenced in Algorithm 4.2. In Figure 4.5 the precision and recall outputs from changing the probability threshold of the detection can be observed. While increasing the probability threshold in the initial stages would eliminate many false positives, the model was not sensitive enough to produce any meaningful changes in recall until the probability threshold went above 0.8. One explanation of this phenomenon could be that the model is very confident in the true positives in the data set, and that only an extremely high probability threshold would eliminate proposal that otherwise be eliminated at a lower threshold.



(a) Suppression of Wiens Output



(b) Suppression of Convolutional Output

Figure 4.4: Non Max Suppression of Output Probabilities

Model	**Recall**	**Precision**
Convolution Network	0.88	0.38
Wiens Features + Support Vector Machine	0.65	0.10
Wiens Features + Logistic Regression	0.65	0.10
Wiens Features + Random Forest	0.80	0.05
Wiens Features + Naives Bayes	0.60	0.10
Wiens Features + Feed Forward Network	0.65	0.10
Wiens Features + Gradient Boosting	0.67	0.10

Table 4.4: On-Ball Screen Detection Results

The methods used by Wiens et al. produced a robust classifier for on-ball screens [32]. Because the input for the Wiens Feature based classifier is constant throughout the entire annotation window, the model will produce a constant probability during detection, seen in Figure 4.4a. With constant probabilities, the detection algorithm with propose a high amount of false positives, thus producing high recall and low precision with proposing new annotations.

Where our approach differs is a non-constant input over the annotation window. We produce image snapshots that show a sliver of the annotation window. These snapshots are entirely different as the movement progresses throughout the window, seen in Figure 4.4b, the model produces a non-constant probability output. When the probability has a range throughout the window, the detection algorithm is able to be more precise with the proposal times and ignore movement during the annotation window that does not resemble the screen moment.

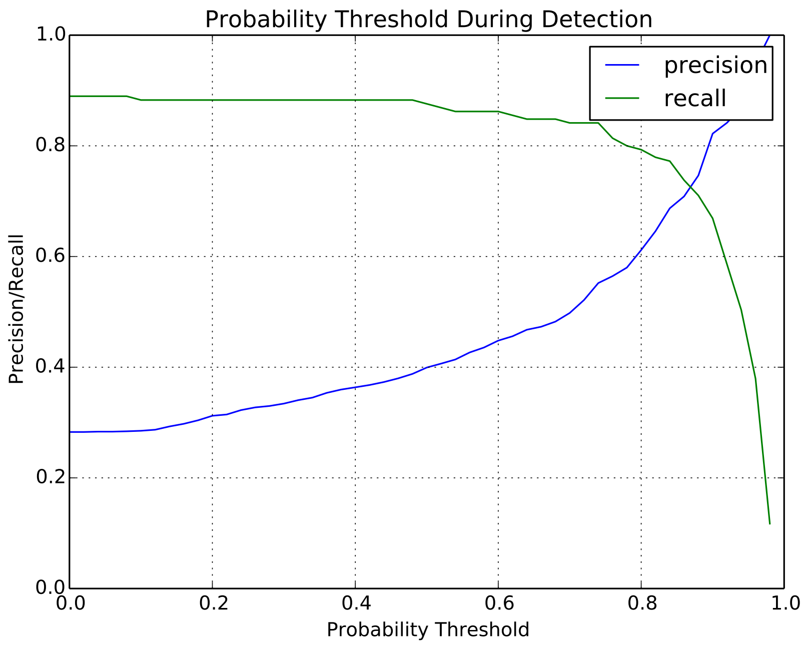


Figure 4.5: Detection Precision and Recall Curve

4.5.2 Proposal Run-Time Comparison

One of the observations that were made when preparing the datasets for the convolution network classifier and the classifier from previous work done by Wiens et al., [32], was that the preparation time for the convolution network was considerably smaller. This can be attributed to the prepossessing that is needed for the Wiens Features described in Section 4.4.1. In the preparation of the Wiens Features, the Euclidean distance between players is calculated n times for all of the movement frames, thus giving a Big O run-time for the feature preparation of $O(n)$. In our proposed method of using raw movement as input to convolutional networks, there is no feature preparation, providing a smaller run time of $O(1)$.

Although the Big O run-time for the Wiens preparation is relatively small, this preparation is done over an entire game of tracking data, which has 72 000 movement frames. We saw a preparation time for a single game go from 20 minutes for our method, go to several hours for the feature preparation of the Wiens Features.

Chapter 5

Unsupervised Learning Methods for Labelling On-Ball Screen Annotations

5.1 Feature Extraction

5.1.1 Behaviour Learning from Trajectories

In topic modelling, a vocabulary of words has to be created to form documents [4]. For the purpose of using topic modelling as a form of unsupervised learning for labelling screen actions, a vocabulary of player actions can be established. This vocabulary should be based on player movement patterns, as well as relational movement to other roles present in the screen action.

Previously, Bornn et al. would use actions identified from trajectory clustering for offensive possession modelling [33]. The actions identified for all ten players on the court were formed from Bezier curve used to represent a players trajectory. The

Bezier curve parameters and control points were then used to create clusters of similar player trajectories. Because they used Bezier curve parameters formed entirely from coordinates, the actions, or clusters identified, were dependant on player location and not on player behaviour. Once player actions were identified, in order to preserve relational context, the actions of all the players at time t were used as the base of single pair-wise actions for each time step.

In work presented by Yao et al., sequences of behavioural features can be used as input to a seq2seq model to find similar trajectories in shipping routes [41]. These behavioural inputs included a feature set extracted from longitude and latitude coordinates, as well as the time spend travelling.

The behavioural features extracted and used as input for action identification are similar to the features presented by Yao et al., such as speed and rate of turn [41]. A sample of three time stamps of player movement can be seen in Figure 5.1, where each data point used can be described by its time from the game clock, its x location, and its y location on the court. These attributes form the basis for the speed and rate of turn calculations.

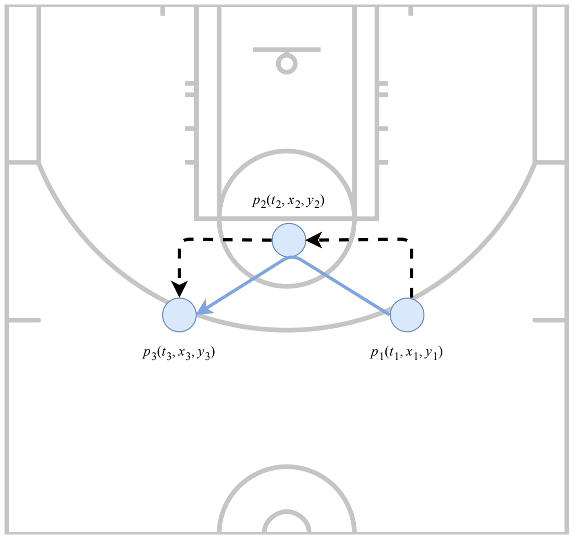


Figure 5.1: Movement Behaviour Attributes

Speed for player p at time i is defined as

$$s_{p_i} = \frac{\sqrt{(x_{p_i} - x_{p_{i-1}})^2 + (y_{p_i} - y_{p_{i-1}})^2}}{t_{p_i} - t_{p_{i-1}}} \tag{5.1}$$

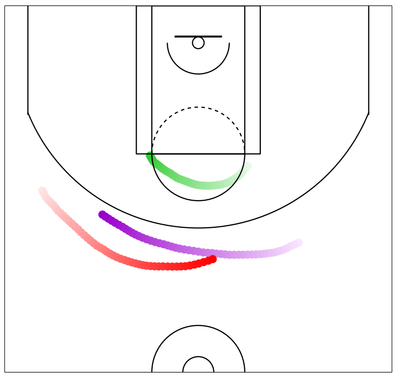
where x_{p_i} is the player p's x location at time i and y_{p_i} is the player p's y location at time i.

Rate of turn for player p at time i is defined as

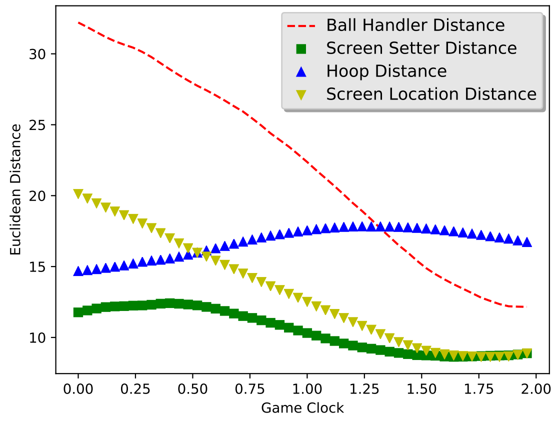
$$r_{p_i} = \arctan \frac{x_{p_i} - x_{p_{i-1}}}{y_{p_i} - y_{p_{i-1}}} \tag{5.2}$$

Domain Specific Features

We also include several features such as pairwise distances between role players involved in the screen as well as the hoop and the screen locations, similar to those used as summary features to describe the approach and execution movement in previous on-ball screen classification efforts described by Wiens et al. [32]. We define domain specific features as the pairwise distance of each of the four role players to the ball handler, the screen setter, the hoop, and the fixed location of the screen. The pairwise distances used as features can be seen for a player for the approach phase in Figure 5.2 and for the execution phase in Figure 5.3.

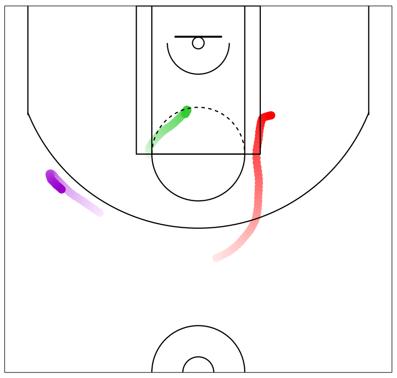


(a) Movement of Screen Defender (Green)

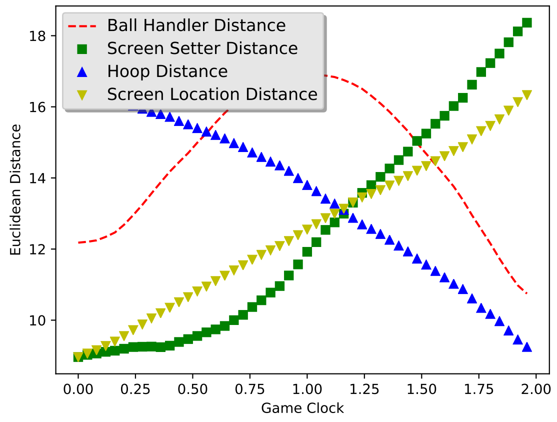


(b) Distances

Figure 5.2: Pair Wise Distances for Approach



(a) Movement of Screen Defender (Green)



(b) Distances

Figure 5.3: Pair Wise Distances for Execution

Sliding Window Extraction

When forming actions from trajectories, it is necessary to form several summaries of the trajectories. Trajectory summaries for the purpose of seq2seq learning can be done by creating behavioural features from a sliding window approach [41]. For each of the windows extracted from a single trajectory, we create summary features on space, time, and multi-agent pair wise distances specific to screen roles to create a description of the movement in the trajectory.

Each feature window formed from the trajectory overlaps with other windows denoted by an offset amount. In our work and in previous work by Yao et al., the offset used is half of the defined window length [41]. The offset is set this way so that each trajectory frame is represented in two windows. In our sliding window feature extraction, our window length is ten frames long and the offset is five frames long, where a single trajectory of length 50 from either the approach or the execution of a screen should have five windows. A visualization of the sliding window is shown in Figure 5.4.

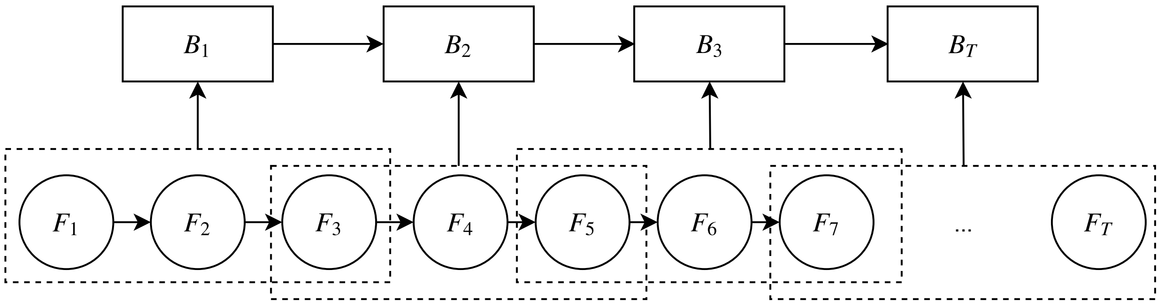


Figure 5.4: Sliding Window Behavioural Feature Extraction

Once the feature windows are extracted, moving behaviour sequences can be formed. With respect to the extracted features, moving behaviours are formed from the differences of the attributes between two records. A window for player p has R records, denoted as $W = \{p_1, ..., p_R\}$, where each frame in the window consists of speed, rate of turn, and the list of pairwise distances described in 5.1.1. The behaviours extracted from the frames are described in Table 5.1.

Feature	**Description**
Change in Time	$\Delta t = t_{p_i} - t_{p_{i-1}}$
Change in Speed	$\Delta s = s_{p_i} - s_{p_{i-1}}$
Change in Rate of Turn	$\Delta r = r_{p_i} - r_{p_{i-1}}$
Change in Distance to Ball Handler	$\Delta bh = bh_{p_i} - bh_{p_{i-1}}$
Change in Distance to Screen Setter	$\Delta ss = ss_{p_i} - ss_{p_{i-1}}$
Change in Distance to Hoop	$\Delta hp = hp_{p_i} - hp_{p_{i-1}}$
Change in Distance to Screen Location	$\Delta sl = s_{p_i} - sl_{p_{i-1}}$

Table 5.1: Behavioural and Domain Feature Input

From previous work in representing behavioural sequences using seq2seq learning [41], each of the features listed in Table 5.1 should be represented in by summary statistics. Where all features $f \epsilon F$ extracted to behaviours B can be represented by its *mean, max, min, 75% quantile, 50% quantile, and 25% quantile.* For each index in the sequence, the entire behaviour window extracted, at time i is comprised of 48 features, shown below.

$$B_i = \{\Delta t, \Delta s, \Delta r, \Delta bh, \Delta hp, \Delta sl\} \times \{mean, max, min, 75\%, 50\%, 25\%\} \tag{5.3}$$

5.2 Methodology

5.2.1 Trajectory2Vec Model

Once feature sequences are formed, we can learn the representation of each trajectory segment from seq2seq embedding. The seq2seq model used for our work in trajectory embedding is comprised of an encoding RNN layer and a decoder RNN layer. Both the encoder and the decoder include a single RNN with 128 hidden units, producing an embedded state for a input sequence of size 5.

When training the model, we train over 300 000 steps, with a batch size of 64, and a learning rate of 0.001 that decays by 0.1 every 100 000 steps. We optimize the model through minimizing the reconstruction loss of the input sequences, defined as

$$Loss = \sum_{t=1}^{n} \left\| B_t - \hat{B}_t \right\|_2 \tag{5.4}$$

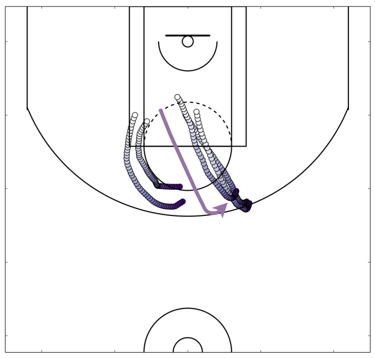
Where the reconstructed output, $\hat{B}$, of the input features B is over the sequence length n, which is determined from the sliding window feature extraction phase.

5.2.2 Action Identification Using Clustering

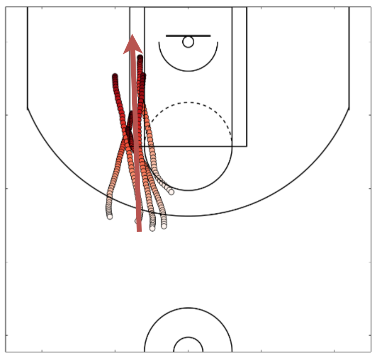
To configure these embedded representations of player trajectories in the approach and execution of on-ball screens as actions, groups of similar trajectories need to be assigned to the vector representations. As output from the Trajectory2Vec model, the embedded representations of the player trajectories are in very high dimensional space (size 128). In order to assign the trajectory representations to groups, the similarities of trajectories to each other needs to be maximized to improve clustering results. In order to reduce the dimensions that the representations occupy and to maximize the closeness of related trajectories, we embed the representations through t-SNE. t-SNE embedding reduces the dimensions of latent representation of the trajectory from 128 to 3, as well as minimizing the distance between related trajectories and maximizing the distance between dissimilar trajectories.

Once the t-SNE embeddings of the trajectory representations are produced, clusters can be assigned to each trajectory. To assign trajectories to groups, we use the k-Means clustering algorithm. In order to ensure all trajectories in documents presented for topic modelling are assigned to action clusters, we applied k-Means. If actions are not assigned to all embedded trajectories, then not all on-ball screens can be represented for the purposes of topic modelling. k-Means is stochastic in nature, but it ensures clusters are assigned to all available trajectories, even though some trajectories may belong to different clusters in new implementations of k-Means.

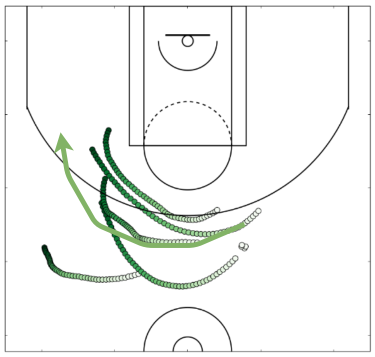
Because the behavioural input does not depend on traditional player location, similar trajectories across the court can be identified entirely based on how fast they turn, how close they get to the ball-handler, etc. Although the trajectories identified in clusters are not dependant on location, for simplicity, we show similar behaviour trajectories that occur in close areas in Figure 5.5. For the actions portrayed in Figure 5.5, we used the behavioural and domain specific features as input for seq2seq embedding, and we identified 50 actions(clusters) using k-Means from the dimensionality reduced representation from t-SNE.



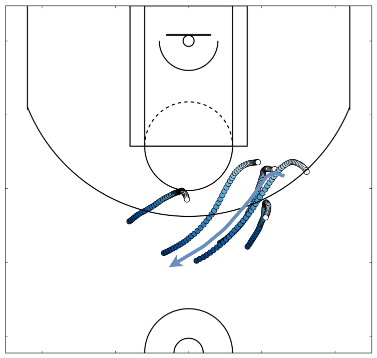
(a) Action 14



(b) Action 10



(c) Action 5



(d) Action 28

Figure 5.5: Actions Identified from Behavioural Clustering

5.2.3 Building Documents from Actions

After identifying player actions from behaviour sequences, documents can be created by representing the actions as words in a text document. In previous topic modelling research, text documents were represented in a Bag-of-Words representation, where counts of each word in a vocabulary were tallied [5], [4]. When representing documents using Bag-of-Words, the context that each word is in is lost.

In detecting defensive schemes in on-ball screens, it is extremely important to know which player is guarding which player [33], [32]. This defensive guarding information can be considered as pair-wise, multi-agent information. In order to preserve this multi-agent relational information, our word representation must preserve context.

Although we can represent an on-ball screen document in a Bag-of-Words context, there is no way for the model to know which specific player is performing which action. In order to represent this context to the topic model, we concatenate the four simultaneous actions for the players involved in a screen into a single pairwise action, similar to the pairwise representation shown previously by Miller et al. [33]. These pairwise actions preserve context for actions, where each action a player takes can be observed in relation to other player actions in an approach or execution window. Because each screen has an approach and an execution, we represent an entire screen as two pairwise actions instead of eight independent actions. The representation of a screen as pairwise actions can be seen in Figure 5.6 and the underlying trajectories that the pairwise actions represent can be seen in Figure 5.7.

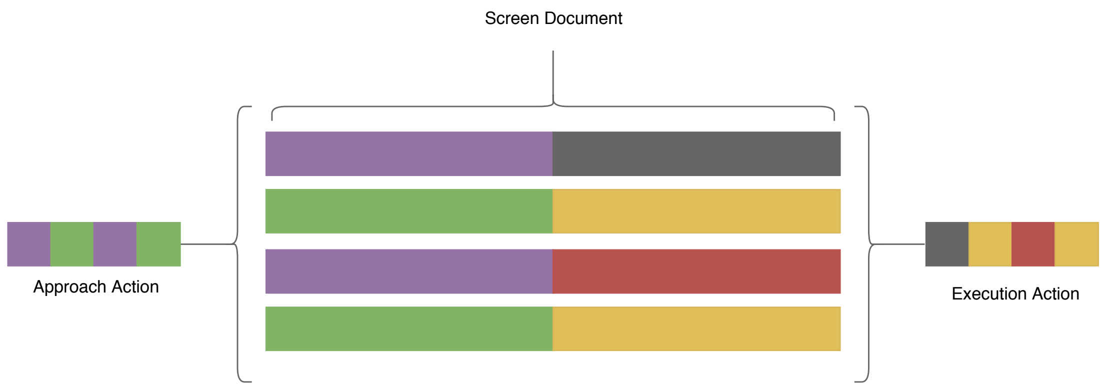


Figure 5.6: Pairwise Action Document

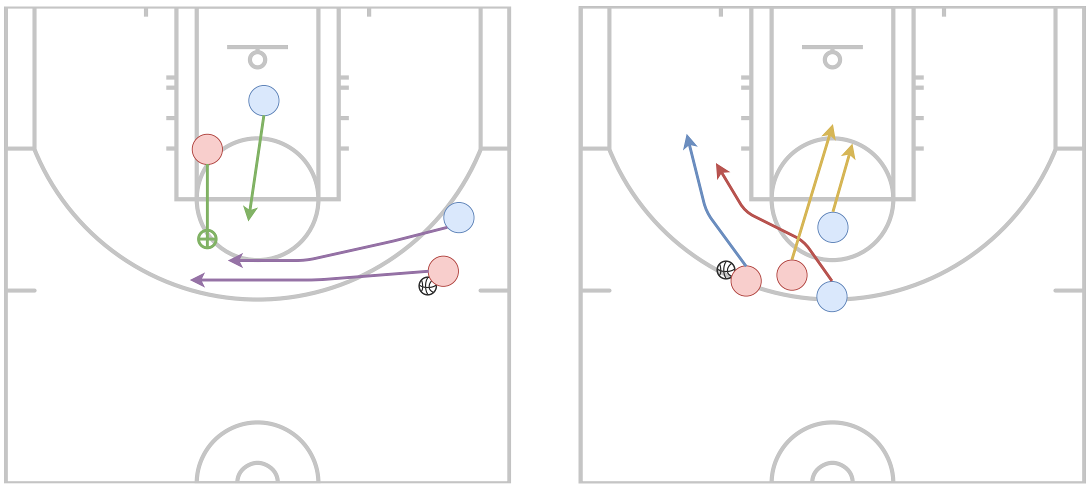

(a) Actions in Approach (b) Actions in Execution

Figure 5.7: Actions Present in Approach and Execution

5.3 Modelling Topics from Screen Documents

Once the approach and execution phases can be represented in a single document, we can form labels or ”topics” from the actions in the document.

These k topics represent the four defensive executions schemes previously shown in Figure 1.6. For each document that is represented by two pairwise actions, the LDA model produces a topic that can be compared to the human annotations given.

For evaluation purposes, we train the LDA model on a corpus of 13 games and single out a single game for validation. The validation set contains a distribution of 39/20/14/20 for the Over/Under/Switch/Trap defensive execution labels in the validation game.

5.3.1 Creating a Baseline Model From Players in Screen

Because there has been no previous work published in unsupervised learning for labelling on-ball screens, we establish a baseline model to show the effect that the actions identified from behavioural and domain specific features have. Without the use of actions identified from behavioural and domain specific features, we can represent a screen document from the players present during the screen. The baseline screen document can be portrayed as a Bag-of-Words representation of the four players involved in the screen. The comparison of the established baseline model to the proposed behavioural and domain feature seq2seq model is shown in Table 5.2.

Model	**Average Precision**	**Average Recall**
Player ID's	0.28	0.31
Traj2Vec NMF	0.33	0.32
Traj2Vec LDA	0.32	0.31
Traj2Vec + Domain Features NMF	0.36	0.35
Traj2Vec + Domain Features LDA	0.37	0.35

Table 5.2: Topic Modelling Results

By creating text documents from actions identified in player movement during an on-ball screen, we can propose defensive execution labels for already identified screens. With the addition of domain specific features as input to behavioural sequence learning, the hidden representations of player movement can be better served in the context of on-ball screens. Using the text documents created from contextual features, we improve the unsupervised classification results of our model compared to the baseline model proposed.

Chapter 6

Conclusions and Future Works

The work presented in this thesis aimed to provide a framework for building a detailed dataset of on-ball screens in basketball using different supervised and unsupervised learning methods. The research utilized data outside standard box scores and outside play-by-play data to provide valuable tools for sports analysts and researchers to create a database of important information on on-ball screens, including labels for defensive execution schemes.

6.1 Conclusions

6.1.1 Preparing a Data Set for Binary Classification

The first problem approached in the work presented was creating a standard on-ball screen data set. In order to create a framework to expand team's data sets of screen annotations, we used a supervised learning approach using convolutional networks. In order to first train our network, we created a set of human annotations of positive and negative screen annotations for the binary classification task. We annotated 14 games worth of screens, enough to produce a trained convolutional model that matches the performance of state of the art models [32], but without the large overhead of identifying domain specific roles to the screen annotations. Without processing these features derived from the screen roles, we saw a large reduction in data preparation time.

6.1.2 Proposing New Annotations Using Detection

Once we had a robust model for binary classification, we were able to utilize it to propose new annotations without any human interaction. The difference in feature preparation between our convolutional model and the models that used the Wiens Features, described in Table 4.1, became ever more noticeable when detecting new annotations. Previous methods require a large amount of processing time for identifying domain specific roles, where the input to the convolutional model did not require any role specific information. Our convolutional network was able to produce variable output probabilities across the annotation window, where previous methods failed to do so. Previous methods used summary features across the entire screen annotation window, thus producing constant probabilities. Because our models produces varying output probabilities over an annotation window, we were able to obtain a higher recall and a higher precision than current models. Using the non-max suppression algorithm, our convolutional model shows promise in producing new screen annotations, as well as minimizing the amount of time needed to process input for proposing annotations.

6.1.3 Creating Multi-Class Labels For Annotations Using Topic Modelling

The second half of the research shown in the thesis was centered around using unsupervised learning to create multi-class labels on screen annotations. This work utilized methods used previously in text processing, specifically topic modelling. We were able to represent a screen annotation as a text document, where trajectory segments in the approach and the execution of the screen can be treated as actions or ”words” in a document.

Traditionally, topic modelling uses Bag-of-Words as input, removing context from words in a document and representing the document as a tally of the unique words. Specifically in the four different defensive schemes or ”topics” we aim to create for the documents, it is very important to understand how players on offense and defense interact with each other. In order to preserve the context in the input to the topic model, we create pairwise actions to represent four players movement as one. By representing a screen annotation as a text document for topic modelling, we can produce multi-class labels in an unsupervised learning setting. Compared to a baseline model without any input from domain feature seq2seq learning or behavioural feature seq2seq learning, our approach produces more accurate results in unsupervised labelling of defensive execution schemes in on-ball screens.

6.2 Limitations

When observing the results of the supervised learning and unsupervised learning applications for the purposes of creating a detailed and accurate data set of on-ball screens, there is a clear difference in results between the two learning types. Our application of convolutional neural networks produced a classifier and detector that would work excellent in a formal setting and could produce accurate predictions compared to other supervised models. For our work in providing multi-class labels using an unsupervised learning model, our results are not accurate enough to replace supervised learning models. Previous work by Wiens et. al has shown supervised learning models to be much more accurate in providing labels for defensive subtypes of the execution of on-ball screens [31]. All though our work is less expensive computationally and does not require human annotations, our model cannot outperform a supervised model and is useless in a practical sense when providing these important details to teams.

6.3 Future Works

6.3.1 Expanding Screen Classification Data Set

In the initial phase of the data set development, we observed the impact that adding more annotated games had to the performance of the model. As we added more games, the validation accuracy continued to improve. We chose not to spend more time annotating more games and instead focused on the detection work proposed in the chapter. We would like to observe the threshold at which the model does not improve it's validation accuracy with the addition of more annotated games.

6.3.2 Comparing Detection Algorithms

In the section of the thesis focused on detecting new screen annotations, our main focus in the research was to show the improvement in precision and recall that the convolution network approach had when creating new annotations. We also described how the approach allowed a faster input preparation time. We did not explore other detection algorithms besides non-max suppression. It would be important for future research to find the best detection algorithm when dealing with raw probability outputs over a window.

6.3.3 Neural Inference Models

The second phase of this thesis was to provide a framework to label screen annotations using the four defensive execution schemes described in Figure 1.6. Although our model outperforms the established baseline model, it does not provide results that would be accurate enough to out-perform analysts in creating these labels. We could improve these results would be done by using more complex deep learning models that can represent the non-deterministic moving relationships players have during a screen. One particular model that has shown promise with inferring relationships between interacting objects in movement data is the neural relationship inference model proposed by Kipf et al. [21]. The model described relies on inferring interactions between moving objects, such as players in tracking data, in a graph neural network structure [26]. Using the inferred relationships created from the neural relationship model, they were able to predict player movement with greater accuracy than their baseline recurrent network model. The movement that was used for trajectory prediction in the inference model came from on-ball screen annotations. Their work shows the impact that relationship inference in an on-ball screen can have on predicting movement sequences. We believe the addition of the relationship inference model would contribute to improving the results of the unsupervised labelling of the screen annotations.

Bibliography

[1] National Basketball Association. stats.nba.com.

[2] Dzmitry Bahdanau, Cho KyungHyun, and Bengio Yoshua. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*, 2015.

[3] Phillip Besse, Brendan Guillouet, Jean-Michel Loubes, and Royer François. Review and perspective for distance based trajectory clustering. *IEEE Transactions on Intelligent Transportation Systems*, 2016.

[4] David Blei, Andrew Ng, and Michael Jordan. Latent dirichlet allocation. *Conference on Neural Information Processing Systems*, 2002.

[5] David Blei, Andrew Ng, and Michael Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993 – 1022, 2003.

[6] Chenjie Cao. Sports data mining technology used in basketball outcome prediction. 2012.

[7] Dan Cervone, Alexander D'Amour, Luke Bornn, and Kirk Goldsberry. Pointwise: Predicting points and valuing decisions in real time with nba optical tracking data. *MIT Sloan Sports Analytics Conference*, 2014.

[8] Kim Esbensen and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2:37–52, 1987.

[9] Felix Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12:2451–2471, 1999.

[10] Lars Hansen and Salamon Peter. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:993 – 1001, 1990.

[11] Mark Harmon, Patrick Lucey, and Diego Klabjan. Predicting shot making in basketball using convolutional neural networks learnt from adversarial multiagent trajectories. *arXiv preprint arXiv:1609.04849*, 2016.

[12] Geoffrey Hinton and Vinod Nair. Rectified linear units improve restricted boltzmann machines. *International Conference on Machine Learning*, 2010.

[13] Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science Magazine*, 313, 2014.

[14] Geoffrey Hinton and Laurens van der Maaten. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[15] Geoffrey Hinton, Ruslan Salakhutdinov, Nitish Srivastava, and Alex Kizhevsky. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[16] Jennifer Hobbs, Paul Power, Long Sha, and Patrick Lucey. Quantifying the value of transitions in soccer via spatiotemporal trajectory clustering. *MIT Sloan Sports Analytics Conference*, 2018.

[17] Kou-Yuan Huang and Wen-Lung Chang. A neural network method for prediction of 2006 world cup football game. *International Joint Conference on Neural Networks*, 2010.

[18] Michael Kates. Player motion analysis: Automatically classifying nba plays. Master's thesis, Massachusetts Insitute of Technology, 2014.

[19] Diederik Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

[20] Diederik Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014.

[21] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Richard Zemel, and Max Welling. Neural relational inference for interacting systems. *International Conference on Machine Learning*, 2018.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *International Joint Conference on Artificial Intelligence*, 2015.

[23] Solomon Kullback and Richard Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[24] Hoang Le, Peter Carr, Yison Yue, and Patrick Lucey. Data-driven ghosting using deep imitation learning. *MIT Sloan Sports Analytics Conference*, 2017.

[25] David Lee and Sebastien Seung. Algorithms for non-negative matrix factorization. *Conference on Neural Information Processing Systems*, 2001.

[26] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *International Conference on Learning Representations*, 2016.

[27] STATS LLC. Sportvu player tracking data. . URL `https://github.com/sealneaward/nba-movement-data`.

[28] STATS LLC. Sportvu player tracking data. .

[29] Stuart Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129 – 137, 1982.

[30] Bernard Loeffelholz, Earl Bednar, Kenneth W Bauer, et al. Predicting nba games using neural networks. *Journal of Quantitative Analysis in Sports*, 5(1): 1–15, 2009.

[31] Avery McIntyre, Jenna Wiens, John Guttag, and Joel Brooks. Recognizing and analyzing ball screen defense in the nba. *MIT Sloan Sports Analytics Conference*, 2016.

[32] Armand McQueen, Jenna Wiens, and John Guttag. Automatically recognizing on-ball screens. *MIT Sloan Sports Analytics Conference*, 2014.

[33] Andrew Miller and Luke Bornn. Possession sketches: Mapping nba strategies. *MIT Sloan Sports Analytics Conference*, 2017.

[34] Andrew Ng. L1 vs. l2 regularization, and rotational invariance. *International Conference on Machine Learning*, 2004.

[35] Adria Sanguesa. Identifying basketball plays from sensor data; towards a low-cost automatic extraction of advanced statistics. Master's thesis, Aalborg University, 2017.

[36] Long Sha, Patrick Lucey, Yisong Yue, Peter Carr, Charlie Rohlf, and Iain Matthews. Chalkboard- ing: A new spatiotemporal query paradigm for sports play retrieval. *International Conference on Intelligent User Interfaces*, 2016.

[37] Rajiv Shah and Rob Romijnders. Applying deep learning to basketball trajectories. *Knowledge Data and Discovery Workshop on Large Scale Sports Analytics*, 2016.

[38] Ilya Sutskever, Oriol Vinyals, and Quoe V. Le. Sequence to sequence learning with neural networks. *Conference on Neural Information Processing Systems*, 2014.

[39] Kuan-Chieh Wang and Richard Zemel. Classifying nba offensive plays using neural networks. *MIT Sloan Sports Analytics Conference*, 2016.

[40] Raymond Wright, Jorge Silva, and Ilknur Kaynar-Kabul. Shot recommender system for nba coaches. *KDD Workshop on Large Scale Sports Analytics*, 2016.

[41] Di Yao, Chao Zhang, Zhihua Zhu, Jianhui Huang, and Jingping Bi. Trajectory clustering via deep representation learning. *International Joint Conference on Neural Networks*, 2017.

[42] Stephan Zheng, Yisong Yue, and Patrick Lucey. Generating long-term trajectories using deep hierarchical networks. *Conference on Neural Information Processing Systems*, 2016.