MIT SLOAN SPORTS ANALYTICS CONFERENCE presented by ESPN 42 ANALYTICS MARCH 4-5, 2022

Using Hex Maps to Classify & Cluster Dribble Hand-off Variants in the NBA

Koi Stephanos, Ghaith Husari, Brian Bennett, Matthew Harrison, Emma Stephanos

Abstract

In recent years, the strategies of NBA teams have evolved with the skillsets of players and the emergence of advanced analytics. One of the most effective actions in dynamic offensive strategies in basketball is the dribble hand-off (DHO). This work proposes an architecture for a classification and pattern mining pipeline for detecting DHOs and their variants in an accurate and automated manner. This pipeline consists of a combination of player tracking data and event labels, a rule set to identify candidate actions, player trajectories embedded into hexbin cell paths, and passing the completed training set of generated features to the classification and clustering models. This work then evaluates the extracted and engineered features, provides a comprehensive accuracy evaluation of the classification models to compare various machine learning algorithms, and finally, analyzes the resulting DHO variant clusters and their ability to provide deeper context to players' actions and abilities.

1. Introduction

With the emergence of read-and-react offenses and the erosion of traditional positions and skillsets, NBA offenses have grown more intricate, frequently relying upon versatile play actions. Advanced analytics so too has evolved to capture the dynamic nature and stochastic decision-making present in these actions [1]. This work explores the increasing complexity of pattern mining action variants in the NBA by constructing a pipeline for analyzing DHOs in SportVU player tracking data [2], rule-based candidate selection, player trajectory hex map encodings, training of classification models and extraction of strategic nuances in execution variants.

1.1 Key Terms

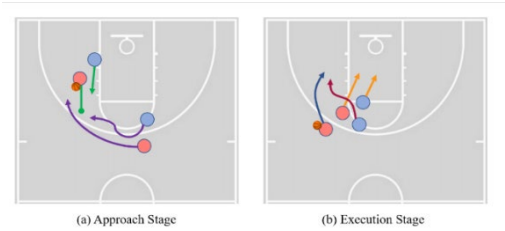
For the purposes of this work, a play is defined as a strategic and intentional sequence of actions carried out by cooperating offensive players in an effort to create the space required for a valuable scoring attempt. An action is defined as a discrete interaction between two or more offensive players and their corresponding defenders. Since only one player may be in possession of the ball at a time, many of these actions involve what is called a screen, in which an offensive player positions themselves firmly between the ball-handler and a defending player, forcing that defending player to navigate around them, subsequently creating space for the non-screening offensive player.

There are many versions of the screen, each with several additional variants, but perhaps the most common is the on-ball screen. An on-ball screen occurs when an offensive player sets a screen for the ball-handler, who then guides their defender into the screen. This can be broken in to two stages: the approach and the execution. The on-ball screen is prominent in many plays and can even be considered a play itself, often referred to by one of its most common variants: the pick-n-roll. The on-ball screen is a favorite in the machine learning literature [3]–[7], as it is a frequently-occurring action with an easily identifiable setup and a variety of possible executions.

1.2 Dribble-Hand-Off

To further explore the potential applications of machine learning to pattern extraction and classification in NBA player movements, this paper targets a less-explored action within the literature: the dribble hand-off (DHO). Much of the prior research into action classification has been focused on the on-ball screen. The DHO, similarly to the on-ball screen, requires two coordinating offensive players in screening and cutting roles, involves an on-ball action often implemented in more complex plays, and can be performed in a number of variants in which either player may attempt to score. As shown in Figure 1, a DHO occurs when the screening player (often a taller front-court player such as a Center or Power Forward) dribbles into a screen and the cutting player (often a smaller player such as a Point or Shooting guard) runs to meet them.

***Figure** 1: Approach and Execution Stages for DHO*

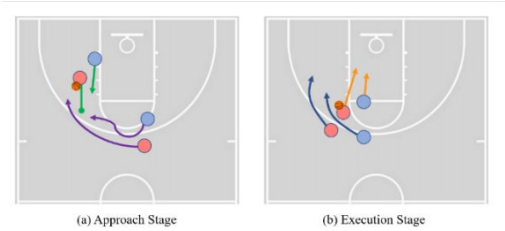


Once the cutting player gets close enough, the screening player "drops-off" the pass and then seeks to become an obstacle for the opponent defending the cutting player. In this example, the screener and cutter and their corresponding defenders all then roll (run) towards the basket. It is interesting to note here that the execution stage in this DHO example is the same as that of the on-ball screen. While both can occur in a significant number of variants in both the approach and execution, the similarities they share in being on-ball actions means they are often implemented in similar situations with similar desired results.

1.3 Fake Variants

Of the many DHO variants, there is one set that requires some additional explanation: fakes. In basketball and other sport strategy, fake plays or actions are often implemented by teams alongside more traditional variants as a way of confusing the defense and creating opportunity. A fake DHO has a similar or even identical approach stage to other DHO strategies, but instead of completing the pass, the screening player will allow the cutter to run under the ball before setting up for the next action or driving to the basket themselves. Figure 2 illustrates a fake DHO that occurs with the same approach stage as the previous example in Figure 1.

***Figure** 2: Approach and Execution Stages for Fake DHO*



The existence of fakes presents a unique challenge in classifying DHOs. When performing an on-ball screen, there are variants where the ball-handling player 'denies' the screen and instead drives in

the opposite direction or the screening player 'slips' on the screen and instead of establishing position, rolls toward the basket. Both variants could be considered fake plays because the typical execution is being diverted in an intentional effort to confuse the defending players, but in both instances, the screen is still set. In the case of the fake dribble hand-off, the hand-off itself does not occur. This can make identification and classification more difficult because the intention of some actions is to divert from the conventional patterns.

Within this paper, fake DHO actions are labeled and grouped with conventional DHOs but given an additional note tag identifying them as a fake. While including these examples likely imposes a lower ceiling on the classification capabilities of the pipeline, overcoming the diversity in strategic variance present in the dataset is a known challenge of this paper and removing those examples would require arguing they are not dribble hand-offs, which cannot be justified. Ultimately, future work to create a separate classifier to identify particular variants, such as fakes, may be necessary.

2. Related Works

Basketball is a team sport in which two teams of five active players compete in timed possessions to amass as many points as possible by scoring, or getting the basketball into their goal, a circular rim positioned 10ft in the air with a glass backboard. Several restrictions are placed on players, who must dribble (bounce) the ball when moving and may not leave the bounds of the rectangular court. In the NBA, games involve as many as fifteen players per team, with twenty-four seconds possessions spanning over four, twelve-minute quarters. Traditionally, a variety of plays are performed in sequence with the objective of creating valuable scoring opportunities. Recently, these strategies have significantly evolved from long-scripted plays to read-and-react offenses in which a variety of independent actions are performed by players who then read the defensive response and react accordingly.

2.2 Machine Learning Approaches

In this section, we create two broad categories to review significant research published on machine learning-based NBA analytics. These are (1) supervised (classification) approaches and (2) unsupervised (including clustering) approaches. In the case of supervised machine learning, a human researcher chooses an event (e.g., an action) that occurs in the data, and then a classification model is built to automatically find all occurrences of that event in the dataset. Unsupervised machine learning, on the other hand, examines the naturally-occurring patterns within the data and analyzes these patterns and their frequencies to potentially reveal new knowledge, groupings, or strategies in the data. While these approaches are different, they are often used together to identify actions and examine general patterns [8]. The key points of each publication are discussed in the following.

2.2.1 Supervised Learning

Supervised learning infers a function from labeled training data to create knowledge structures that support the process of classifying new instances into a set of predefined classes [9]. The input for this type of learning algorithm is a collection of sample instances that are pre-classified (labeled) into a predefined set of classes. The output of this process is a classification model that is constructed by analyzing the training data and producing an inferred function that can determine the class (label) for unseen instances with a reasonable accuracy.

Due to their top-down nature, supervised learning efforts in NBA analytics are often more focused on identifying and labeling certain events in the dataset. Specifically, in the case of the on-ball-screen, research works formulate and utilize a set of rules around the distances and durations of time that coordinating offensive players and the ball spend in proximity to each other. Approaches that use a sliding window to extract features of a certain event are frequently used to ensure the capturing of all actions, as it is difficult to specify the point in a given possession at which an action has started [10]. Using a sliding window, the screen moment can be specified by identifying the moment right before the players begin moving again [4].

Yu and Chung [11] propose a binary classifier to automatically identify on-ball screens. First, they propose an algorithm to identify event candidates that may contain on-ball screens. This algorithm uses a set of rules that are devised manually to extract on-ball screen candidates and filter out other events. These candidates are then analyzed manually (by humans) to determine and label each as on-ball screen or otherwise. Once the labeled dataset (actions and their labels) is created, the set is split into two parts: a training set, and a testing set. This split is often 9:1, where 90% of the data is used for training and 10% is used for testing. The SVM binary classifier was trained using four distance- and speed-based player features. After the training phase, the testing set was fed to the classifier to evaluate its performance identifying on-ball screens, and it achieved a recall of 90.46%.

McQueen and Guttag [4] propose an approach that utilizes a set of rules to identify actions. The data are then segmented into periods of time around these actions. Using this approach, they extracted 30 continuous features from each action. Once extracted, these features were discretized into five binary features based on quintiles. This resulted in a 150-dimensional feature vector for every action. The data were then labeled and split into approximately 51.9% training and 48.1% validation. A linear classifier was then constructed using the training data. This process was repeated 3,200 times using different splits of data. The classifier achieved a recall of 82% and precision of 80%.

McIntyre et al. [3] use a more robust dataset consisting of 270,823 ball screens. The labeled dataset was split into 70% training and 30% testing sets. Four classifiers were trained to identify four different types of ball screen defensive strategies. These classifiers identified 270,723 screens in total, achieving a high recall of 83% and precision of 78% in identifying the type of on-ball screens where the defender stays between the ball handler and the screener.

Another approach proposed by Kates [10] uses the SportVU data to detect six different plays, where each play has 20 labeled instances at minimum. The approach trained six SVM classifiers (one for each play type) using an 8:2 stratified data split. The classifiers predicted the plays with a collective accuracy of 72.6% and an F-score of 72.7%.

Wang and Zemel [12] use variants of neural networks to automatically classify play sequences to 11 selected play classes (or offensive strategies). Recurrent Neural Networks (RNNs) achieved the highest accuracy , top-3 accuracy of 80%, in classifying 95 unlabeled sequences (approximately 6% of the data) into 11 possible play classes. Predictably, the simple neural networks achieved a lower top-3 accuracy of 77% when compared with the RNN. This is due to the ability of RNNs to better handle and learn from sequential data of variable length as it accumulates change over time.

2.2.2 Unsupervised Learning

Unsupervised learning (specifically clustering) is a type of machine learning that allows the model to work with minimal or no human supervision by finding the natural clusters (groups) in the data using heuristics without reference to labeled outcomes by humans [9]. Clustering focuses on finding patterns in the data and creating groups of instances with similar properties. This similarity is calculated by a distance function, such as Euclidean distance. Depending on the approach, these clusters can be exclusive, in which instances belong to only one cluster, or they can be overlapping, where an instance could belong to more than one cluster.

Unsupervised learning is becoming increasingly popular among sports scientists. Research works that utilize unsupervised learning methods are more geared towards pattern extraction and strategy analysis than supervised efforts. However, they often leverage the classification potential of supervised models to do so.

Nistala and Guttag [1] and Nistala [13] utilize cluster analysis to assign group numbers to similar attacking movements, such as movements along sidelines, runs along the baseline, and other attack movements. First, they construct 3 million trajectory images from NBA player tracking data collected by the STATS SportVU player tracking system [2]. They then utilize a numerical abstraction algorithm for player movements and run the K-means clustering algorithm on the 3 million trajectory images to group similar trajectories together. The algorithm grouped the attacking player movements into 20 clusters. Manual evaluation of these clusters (by selecting case studies from each one) showed that deep learning can be used to learn patterns of basketball attack movements.

Brooks [6] proposes an approach utilizing unsupervised machine learning to characterize patterns of play for NBA teams on offense. First, the approach builds an image for each player's movement on offense to give a starting point for comparing player movements across different possessions. They then utilize the k-means clustering algorithm using the Python package scikit-learn [14] to cluster 60,000 instances of possessions. The number of clusters was set to $k = 30$, as it represents the ``knee'' in the response curve between k and the average within-cluster distance. This process produced 30 clusters, each containing around 2,000 instances. The 30 resulting clusters were evaluated manually (using human judgement) by checking the top ten instances that are closest to the center of the cluster. This evaluation confirmed that instances in the same cluster had similar patterns of movement for the players and the ball. Each cluster was then given a description that characterizes the movement pattern (or play) of its instances.

Lutz et al. [15] propose an approach that uses statistics such as field goals and assists to cluster similar players into ten categories. Franks et al. [16] utilizes non-negative matrix factorization (NMF) to cluster defense players based on the locations of field goals. Sampaio et al. [17] and Kalman and Bosch [18] cluster players with similar features such as attacking, defense and passing statistics. Sampaio et al. [19] and Teramoto et al. [20] apply dimensionality reduction techniques such as principal components analysis (PCA) in basketball.

Since 2005, many NBA video analytic research works have been proposed that have motivated and impacted the domain. In this section, we summarize the key challenges identified in the literature and highlight proposed solutions and directions for future research. We divide the challenges and proposed direction into three categories: (1) feature engineering and extraction, (2) deeper information discovery of actions, and (3) effective video content preprocessing and noise filtration.

Despite the many research works and proposed solutions for action and play recognition in NBA data, many problems remain due to the complexity of recognizing and differentiating between actions and sub-actions of basketball plays. It is challenging to develop a unified framework with a machine learning pipeline that can filter, abstract, and label actions and plays in an accurate manner.

3. Data and Event Labels

In 2012, the NBA began tracking player movements during games by installing a total of six cameras that capture the location of all ten players on the court, as well as the ball and three referees, 25 times per second. This data was publicly available through the 2015-2016 season and inspired a great deal of interest among the sport analytic community, eventually making its way to machine learning experts [2]. The data itself is collected in a JSON object, typically around 600 MB in size, containing a variety of defining information such as the teams and players involved, the date and location of the game and the coordinates of the players broken up by quarter and event.

An event occurs when one team takes possession of the ball and ends either when they score, a foul or turnover is committed or in other unique circumstances, such as clock malfunction. Since the raw positional data of the players does not paint a full picture, this dataset is often combined with other event-specific data, such as the type of shot taken or other play-specific details that allow for a more cohesive view of on-court behaviors. Many techniques are used to abstract this low-level data, but most of these include the embedding of the coordinates into vectors or trajectory images that represent a player's full range of motion over the course of a given possession.

To provide additional context and higher-level descriptions to the raw coordinate data, event labels are loaded and associated with the SportVU events by their corresponding event numbers. These event labels were originally created by researchers in [21] and a summary of them can be found in **Table** 1.

***Table** 1: Summary of Event Labels*

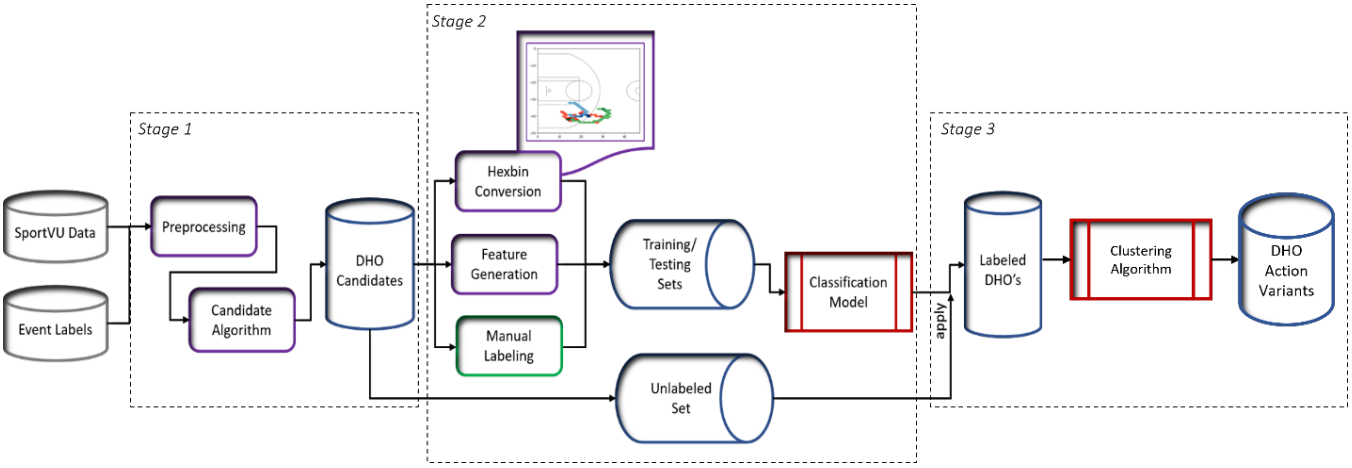
Column Name	Description
EVENTMSGACTIONTYPE	type of action (enum)
EVENTMSGTYPE	further breakdown of action (enum)
EVENTNUM	event number/order in game
GAME_ID	NBA game id
HOMEDESCRIPTION	raw textual description of event for home team action
NEUTRALDESCRIPTION	raw textual description of event if neutral action
VISITORDESCRIPTION	raw textual description of event for away team action
PERIOD	current period
SCORE	game score
SCOREMARGIN	difference of game score (home - away)
WCTIMESTRING	real world time
PCTIMESTRING	time left in quarter
PLAYER1_ID	NBA player id for player 1 (in play-by-play description)
PLAYER1_NAME	player 1 name
PLAYER1_TEAM_ABBREVIATION	team abbreviation for player 1
PLAYER1_TEAM_CITY	team city for player 1
PLAYER1_TEAM_ID	team id for player 1
PLAYER1_TEAM_NICKNAME	team nickname for player 1
PLAYER2_ID	NBA player id for player 2 (in play-by-play description)
PLAYER2_NAME	player 2 name

PLAYER2_TEAM_ABBREVIATION	team abbreviation for player 2
PLAYER2_TEAM_CITY	team city for player 2
PLAYER2_TEAM_ID	team id for player 2
PLAYER2_TEAM_NICKNAME	team nickname for player 2
PLAYER3_ID	NBA player id for player 3 (in play-by-play description)
PLAYER3_NAME	player 3 name
PLAYER3_TEAM_ABBREVIATION	team abbreviation for player 3
PLAYER3_TEAM_CITY	team city for player 3
PLAYER3_TEAM_ID	team id for player 3
PLAYER3_TEAM_NICKNAME	team nickname for player 3

4. Design and Implementation

The pipeline presented in this paper is designed to take in positional data and event labels representing the on-court actions over the course of a particular NBA game and produce a training set and model capable of classifying particular action instances. To accomplish this, a sequence of preprocessing steps to combine and filter noise from the datasets was performed. The relevant games, players, teams, events, and moments are then serialized and stored in a relational database. To account for the variant nature of NBA actions, the pipeline implements a candidate algorithm to cast a wide net and collect as many action instances from the raw data as possible. These candidates are then reviewed manually by watching live game recordings. The positions of the screener, cutter and ball are then encoded at points of interest using hexbin maps and are included with the other extracted and engineered features. Once the features are complete, the resulting dataset can be split and used for model training and testing. A visualization of this process is found in Figure 3. Following is a more detailed description of each phase of the pipeline.

***Figure** 3: DHO Classification and Pattern Mining Pipeline*



4.1 Preprocessing

In the preprocessing stage, the two datasets are loaded in from their respective JSON and .csv files and combined by event id. Data for the teams and players involved in the game, such as the names and positions of players and the team cities and abbreviations, is then extracted and used to create the corresponding database entities for reuse.

Once the descriptive data is extracted and the datasets are combined, the events are narrowed to only those related to a successful or failed scoring attempt, a personal foul, or a turnover. This is because many of the events contain no corresponding positional data or because those events do not contain moments of interest in which a DHO or other action might occur. Examples include the events marking the start and end of a quarter and those corresponding to free throw attempts. Once this step is completed, the other non-pertinent data columns are removed, and event ids are generated so they can be stored in the relational database.

To complete the preprocessing steps, several higher-level features are extracted from a combination of positional data and event labeling to enable our candidate algorithm to detect potential DHO actions. First, it must be determined which team is in possession of the ball for each given event. The raw coordinate data often extends well before and after the event in question, even overlapping neighboring events. This makes it difficult to tell which sections of the coordinate data ought to be ignored for a given event. Once possession has been determined, directionality can be established by looking to see which basket each team is attempting to score on (this then flips during the second half of play). When the possession and directionality have been confirmed, the moments for the event can be extracted from the raw data.

Throughout the preprocessing and subsequent processes, several cases of inconsistencies that could not be reconciled were discovered. An example of this is when a tracking camera switches identifications or loses a player entirely, resulting in players jumping wildly between two different locations. Other discrepancy cases where the clock suddenly reverts or becomes constant during play or even suddenly becomes a series of "NaN" values make it impossible to confirm based upon the raw data what exactly happened during the event. Considering the dataset is publicly available and from 2016, this paper assumes that many of these inconsistency issues in the raw data collection have been addressed and are therefore of little academic interest to the machine learning applications present in this pipeline. Therefore, odd events containing egregious raw data inconsistencies were removed from consideration.

4.2 Extracting Candidate Dribble Handoff Events

As discussed previously, a rule-based algorithm can identify and extract basic actions from the complex, real-time player motion data. For example, a screen is often identified in the literature by calculating relative distances for all the offensive players on the court and looking for any pair whose relative distance is within a certain margin, often five feet. Obviously, such an approach is somewhat limited, because it is often the case that two players are within a five-foot proximity of each other, but no screen occurs. They may be jockeying for rebounding possession, running past one another, or simply standing closely. This makes classifying any action, even one as simple as a screen, a complicated task that requires either many specific rules that run the risk of overfitting the data or a small set of looser rules that either capture too much noise or fail to identify numerous positive instances.

Due to these challenges, relying only on rule-based algorithms to identify and classify classifying complex actions involving multiple coordinating players provides limited outcomes, especially in terms of precision. The candidate algorithm is therefore designed with a minimal rule set intended to capture a wide training set of both positive and negative instances so that the actual classification can be performed by a machine learning model informed by the actions that pass the candidate algorithm's rules.

4.2.1 Identifying Passes

The first step in the DHO classification pipeline is to identify the player in possession of the ball at a given time. In basketball, this individual is referred to as the ball-handler and can be found by taking the Euclidean distance from the ball to each of the offensive players and choosing the player corresponding to the minimum distance [11]. While this is not perfect and cannot capture certain situations, like a player fumbling a ball, it is a reasonable start given the constraints of the dataset. A few additional rules concerning the radius of the ball are added to help rule out instances where a shot is taken, or players are fighting for a rebound and no one is technically in possession. Finally, a distance threshold of five feet is set, and for all moments in which no offensive player is under that threshold, possession is marked as "N/A".

From the ball-handler data, a list of all passes for a given event can be extracted by looking for shifts in possession. Since the moments when possession is lost and regained are marked and associated with a specific coordinate in the SportVU data, this data can be used to set the start and end locations of the pass and determine how long it took for the pass to be completed.

4.2.2 Selecting Candidates

Finally, the list of extracted passes is combined with two additional rules to eliminate certain types of passes, such as those that occur in the painted area closest to the basket (the paint) or those that originate from out of bounds. While it is not impossible to have a DHO that occurs in the paint, these additional rules filter a significant amount of noise from the final dataset, while losing only a handful of positive instances.

The passes are then evaluated by how long it took for the pass to be completed. This is measured by looking at when the ball first leaves the possession of the passing player and the corresponding moment in which the ball first enters the possession of the receiving player. Any remaining pass that occurs within a small enough time span (typically eight moments, depending on the capture rate for that particular game) is nominated as a candidate. The candidate is then assigned an identifier and a label, and a list of identifying features is created for the manual labeling process before it is added to the relational database.

4.2.3 Evaluating Selected Candidates

Of the resulting 3398 candidates selected across the forty-three games reviewed in this paper, a total of 1098 was identified as positive DHO instances. A portion of the positive DHOs, ~5.2%, were marked as fakes, which are discussed in depth in section 1.3. A summary of the breakdown for the labeled candidates, as well as the associated source data, can be found in **Table** 2:

***Table** 2: Overview of Source Data and Extracted Candidates*

Data Overview			
Games	43	**Total Candidates**	3398
Teams	29	**Positive Candidates**	1098
Players	404	**Negative Candidates**	2300
Events	2406	**Fake Candidates**	57

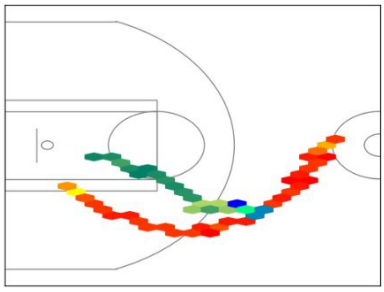
Ultimately, the intent of the candidate algorithm is to cast a wide net and collect as many action instances as possible that can later be narrowed by the trained classification model. Therefore, time was not spent further engineering or optimizing the rules that dictate the nomination process. A total recall of ~90% was achieved during manual labeling, and this was deemed sufficient to train the model. Additional steps to increase recall or narrow the scope of candidates can be considered in future work.

4.3 Hexbin Positional Conversion

One challenge in identifying patterns in such a diverse dataset is overcoming the granularity and precision of the raw data. Each positional coordinate in the SportVU moments contains two floating point numbers with as many as eight trailing digits. Feeding this raw data into a classifier can be confusing, as two nearly identical positions may evaluate to have low similarity scores.

To abstract away some of these play/action details and capture the higher-level descriptive elements of players' court positions at key moments during an action, this paper uses a technique known as hexbin mapping to embed player trajectories into cell paths. Binning occurs by separating continuous data into discrete sections, in this case hexagons. The raw coordinates are then placed into each one of these bins, and the resulting occupancy counts of those bins can be used to represent the original data. An example of hexbin mapping seen in Figure 4 shows how this strategy can be used to convert the trajectory paths for the screener (greens), cutter (reds) and ball (blues). A color gradient is used to indicate the rate of player movement, such that the more positions that are recorded in a specific cell, the darker on the gradient that cell appears.

***Figure** 4: Hexbin Encoding of DHO Action*



4.4 Feature Generation

A selective list of 50 features was generated for each candidate. A total of 16 of these features are engineered, in that they require significant manipulation or composition past the point of a simple calculation. The remaining 34 features were extracted from the raw dataset and may or may not have gone through one or more processing steps amounting to basic calculations. Once all the features were collected, they were combined with the manual label, given a derived id, and stored in the relational database for use in training the classification models. Both the extracted and engineered features are discussed in detail below.

The features generated were derived from two-second windows before and after the screen moment, defined as the moment when the coordinating offensive players are closest together. By

distinguishing the approach and execution stages within the feature set, examples with vastly different executions can still be classified similarly by the classifiers.

4.4.1 Primitive Feature Extraction

In total, 42 of the features used to train the classification models for this paper were extracted from the combined SportVU and event labels dataset. Most of these extracted features required some basic processing to derive meaningful metrics from the data before being passed off to encoding and normalization steps. The exceptions to this are the player archetype and the ball radius, which is a floating-point measure of the ball size on the camera that indicates the approximate height of the ball, where larger radius measures indicate a greater height along the z-axis. The ball radius is left as a floating-point value and passed along to the normalization process, but the player archetypes require some form of encoding because it is difficult for a classification model to compare string values. Encoding the player archetype designation creates an enumeration, which in turn can be represented by an integer and accurately utilized by the learning model.

The remaining 36 extracted features consist of various measures of the paths traveled by the cutting and screening players as well as the ball itself. The first of these are the distances each traveled, as well as the relative distance of the players at key moments of considerations: the beginning of the approach stage, the pass moment, the screen moment, and the end of the execution stage. These are the same moments for which the engineered location features discussed in the following subsections will be calculated. In addition to the distance measures, the average speed of the cutter, screener, and ball, as well as the slope and trajectory of their plotted movements over the action duration, are derived from the raw data. A full overview of these features and their descriptions can be found in Table 3 below.

Table 3: Overview of Extracted Features

Feature	Feature Type	Feature Description
screener archetype	Enumeration	Encoded string containing players' assigned positions
cutter archetype	Enumeration	Encoded string containing players' assigned positions
distances traveled*	Float	Calculated for the screener/cutter/ball
average speeds*	Float	Average speed of screener/cutter/ball
trajectory slopes*	Float	Slope of screener/cutter/ball trajectory
trajectory intercepts*	Float	Intercept of screener/cutter/ball trajectory
ball radiuses**	Float	Radius of the ball indicating relative height
relative distances**	Float	Relative distance between the screener and the cutter from each other and from the ball
**** represents multiple features collected for the cutter/screener/ball from the approach and execution stages***		
***** represents multiple features collected from the start/end of the action and the screen/pass moments***		

4.4.2 Composite Feature Engineering

In this paper, a total of 16 features required significant enough abstraction or data manipulation that they were considered to be engineered. The first of this set, the hexbin cell transform of the location data, has already been discussed in detail. While the other engineered features in this paper would likely be found in sophisticated classification efforts for different actions, they have not, to the knowledge of this paper, been used to distinguish DHO candidates.

These features consist of higher-level composite features that, when combined with the raw coordinate data and a rule-based algorithm, contribute a deeper contextual and temporal understanding of the on-court behaviors present in the action. The first of these measures is the

offset into the developing play itself. This is based off of the insight that coordinated offensive actions typically require time to set up, execute, and then create a scoring opportunity. This means such actions are less likely to occur at the very start of the possession, when the majority of offensive players are likely still on the far side of the court, nor at the very end of a possession, when players are scrambling to get up an attempt before the shot-clock expires. Similarly, all ten player locations are considered to determine how many are on the half of the court where the team in possession is currently trying to score. If only seven of the ten players are past half-court, the players are likely in a fast-break offensive situation in which each player is darting towards the rim or three-point line, and the potential for any type of on-ball action is significantly limited.

Finally, in addition to the duration of the pass (derived from the pass detection algorithm described in the candidate algorithm), the initial pass is examined to detect if it originated from out-of-bounds, which indicates what is referred to as an inbounds play. Inbounds play are unique offensive opportunities in basketball, in which players and often coaching staff are permitted to gather and discuss strategy ahead of the proceeding play. This opportunity for verbal collaboration commonly results in a higher potential for coordinate team action and, therefore, an increased rate of play-actions such as DHOs. Ultimately, these engineered features are calculated for each candidate alongside the extracted features and combined along with an id and manual label to create the final testing/training set. An overview of each of these engineered features can be found in Table 4 (16 total).

***Table** 4: Overview of Engineered Features*

Feature	Feature Type	Feature Description
cutter locations*	Hexbin coord.	Coordinate of closest hexagon cell for cutter on the start/end of the action and pass/screen moments
screener locations*	Hexbin coord.	Coordinate of closest hexagon cell for screener on the start/end of the action and pass/screen moments
ball locations*	Hexbin coord.	Coordinate of closest hexagon cell for ball on the start/end of the action and pass/screen moments
offset into play	Integer	Indicates which sixth of the play the action occurred in
pass duration	Integer	The total number of moments captured between the start and end of the pass
players past half court	Integer	Count of players currently on the same side of the court as the occurring action
inbounds play	Boolean	Indicates whether play is initiated from an out-of-bounds pass
**** represents multiple features collected from the start/end of the action and the screen/pass moments***		

4.4.3 Predictive Machine Learning Model Construction

Our approach's final step is to construct a machine learning model to classify the generated images into either covert or overt. For this task, we train a set of models using the following machine learning algorithms:

- Support Vector Machine (SVM)
- Decision Tree (DT)
- Gaussian Naïve Bayes (GNB)
- Artificial Neural Network
    - Keras w/ TensorFlow (Keras)
    - Multilayer Perceptron (MLP)

We trained each classification model using the features extracted from each image. Then, we evaluated the models using the hold-out validation method. Based on the hold-out validation method, we split our dataset into 70% for training and 30% for testing (validation). To construct and validate these models, we use the popular data mining tool: Waikato Environment for Knowledge Analysis (WEKA 3.8) [22].

4.4.4 K-means Clustering Formation and Selection

4.4.4.1 Precision Abstraction

In other approaches in literature, a convolutional neural network (CNN) was used in place of a hexagonal cell path encoding as a form of precision abstraction [1], [12]. A CNN uses multiple convolutional and deconvolutional pooling layers to "fuzz out" an image, creating a more abstract representation that is more easily digested by machine learning algorithms. This is because by removing small, idiosyncratic details from an image, more emphasis is placed on higher-level trends.

One contribution of this approach is that it is a potential alternative to using a CNN. While effective, there is no particular intention behind the abstraction of certain details. By capturing player movements in cell locations, the same goal of precision abstraction is achieved, but there is also a clear explanation as to why a particular trajectory appears the way it does. These cell locations could even be used individually to refer to specific areas or locations that hold particular in-game value, much like coaches or players would refer to specific locations such as "the low-block" or "the left elbow". Ultimately, there is the potential for future work to combine hexbin encodings with a CNN.

The resulting image encodings of the player and ball trajectories must then be prepared for the clustering process. This requires converting the images to numerical vectors that can be ingested by the machine learning algorithm. The images themselves are 480x640 pixels, and occupy three channels for red, green, and blue. This data is then compressed into one long series of nearly a million total individual pixel color weights.

Considering the prevalence of white space in the images, a considerable amount of the final data is white noise. For this reason, principal component analysis (PCA) was used to trim data with little to no variance. This is done by analyzing the entire compressed dataset and identifying the dimension by which the most variance within the data is preserved while eliminating those sections that provide little to no differentiation between samples. The PCA algorithm was able to reduce the 921,600 initial pixels to only ~573. This dramatic reduction allows the clustering algorithm to compare the samples and produce tight and distinguished clusters more easily.

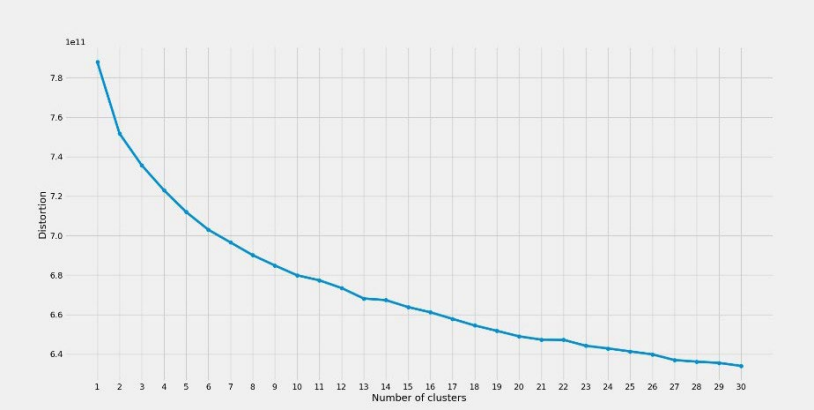
4.4.4.2 Determining Number of Clusters

Once the data has been fully prepared, it is ready to be clustered. In this paper, we used the scikit-learn python libraries implementation of k-means clustering [14]. K-means is a nearest neighbors clustering algorithm by which samples in a dataset are evaluated based on similarity scores and placed into corresponding groups based on the perceived closeness in their appearance. This unsuperivised learning algorithm is ubiquitious in the literature, due in large part to its effectiveness at detecting patterns and variants in datasets that may have been initially unknown to the researches themselves. Considering the dynamic enviornment in which NBA actions occur, this makes k-means a strong candidate.

Two metrics were used to determine the quality and appropriate number of clusters: distortion and silhouette score. Distortion is a measure of the overall tightness of the clusters (how densely packed the samples belonging to the cluster are), while the silhouette score represents the distinction between clusters (the space between the end of one cluster and the beginning of another).

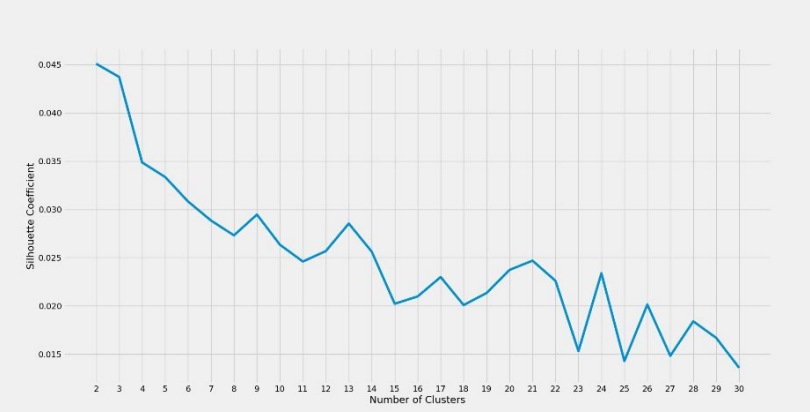
A distortion plot is often referred to as an elbow plot, because in an ideal dataset, there is a sharp and noticeable bend at the ideal number of clusters [23]. For a diverse dataset, this is not always as obvious, but by examining how distortion drops as the number of clusters is increased, a trend as to where the diminishing returns begin can be identified and used to target an ideal number of clusters. As seen in Figure 5, there is no clear elbow, but it does appear as though the algorithms begins to experience diminishing returns between clusters 9-11.

***Figure** 5: Elbow Plot for K-Means Clusters*



Looking at the silhouette scores, we can see a clearer local maximum at cluster 9. This indicates that in this range we have a good balance between the tightness and separation in our clusters, which represents a good fit. Fewer clusters run the risk of grouping dislike actions together and more clusters could potentially separate actions that are closely aligned and would be more reasonable categorized together. The silhouettes scores for clusters 2-30 are shown in Figure 6.

***Figure** 6: Silhouette Scores for K-Means Clustering*



Upon reviewing the above discussed metrics, a cluster count of nine was chosen, which we believe strikes a good balance between cluster density and separation and will result in the most meaningful insights.

5. Evaluation and Analysis

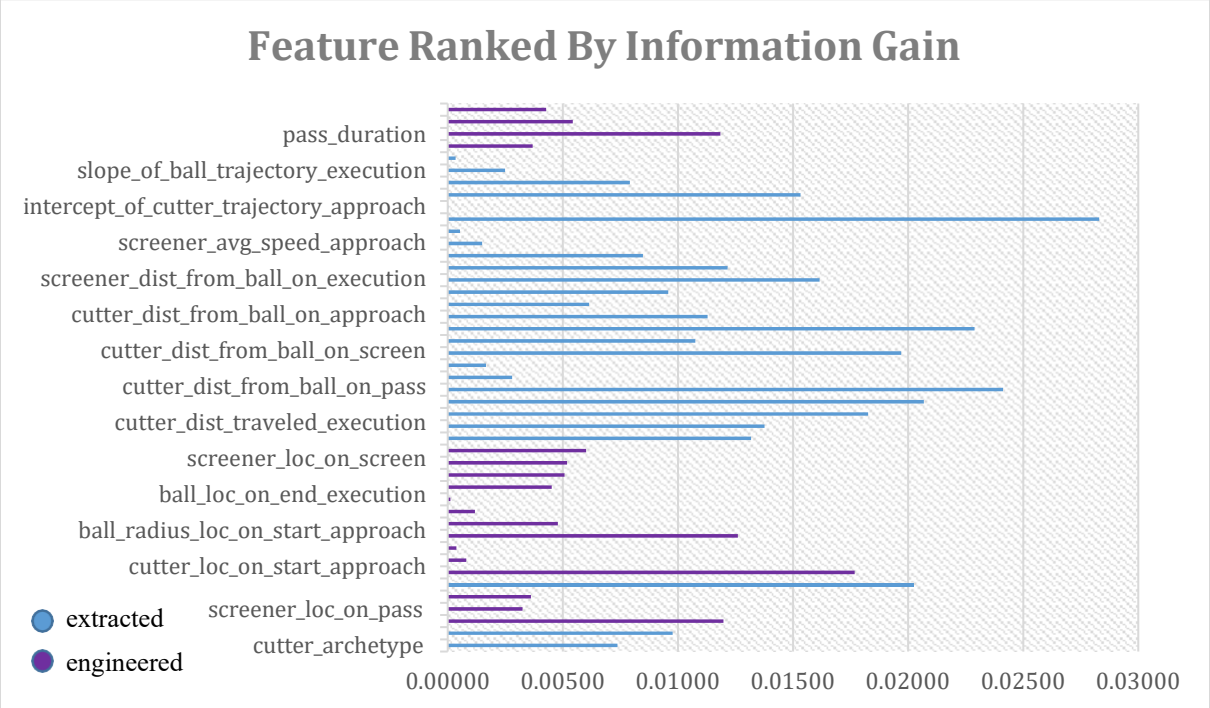
5.1 Accuracy Measures

To evaluate our approach, we use several performance metrics to investigate its accuracy and completeness under different settings. The most popular accuracy measures in machine learning and information retrieval domains are accuracy, precision, recall, and F1 score. In addition to these metrics, receiver operating characteristics (ROC) were generated for each of the learning models. An ROC is used to compare the false positive and true positive rates of a model. A completely random model with an even data distribution tasked with choosing between two classes can be expected to do so with approximately 50% accuracy. If plotted, this would result in a diagonal line at a 45-degree angle extending out from the origin. The area under that curve (AUC) would be 0.5, or approximately 50% of the domain space. A model that efficiently distinguishes between class designations, on the other hand, would perform better and, in turn, have a higher AUC. It has been found that an AUC score of closer to 1 not only indicates a highly performant and predictive model but that such a measure is even more telling than accuracy metrics [24].

5.2 Features Effectiveness

There are several metrics available to evaluate the quality of a feature set. Considering that the effort of this paper is to build a classification model, information gain is an attractive method, as it ranks features in relation to some targeted dependent variable. Information gain is determined by calculating the mutual information of each variable in the feature set with respect to the classification based upon the entropy of those elements [25]. The resulting rankings indicate which features have the most predictive power within the dataset. A summary of the feature rankings for this paper is found in Figure 7.

***Figure** 7: Information Gain for Proposed Features*



5.3 Experimental Results

To evaluate the effectiveness of the constructed pipeline and composed feature set, the training data was first fit on and subsequently evaluated by three classification learning models implemented by the Scikit-learn library: SVM, GNB, and DT. SVM is a favorite among the literature for the classification of other on-ball actions and shows promising potential in classifying DHO candidates as well. All models in this paper were configured independently and then assessed by averaging results from 1000 distinct iterations with unique testing splits to avoid placing too much stake in any individual model. As seen in Figure 8, SVM outperforms the other two learning models on precision, recall and f_1 score

5.3.1 Classification Models

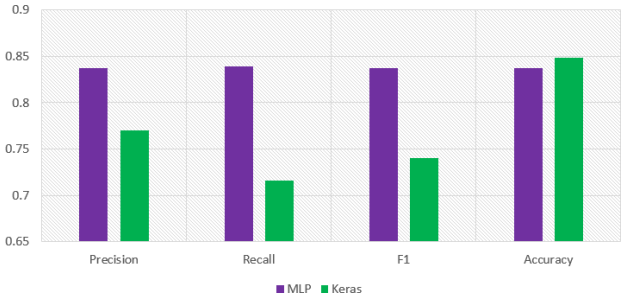
While these indicators seem to suggest that SVM is a favorite for classification among these options, another measure of a model's fitness, the ROC curve, offers additional insight into how well the model discerns between the classifications. This is important, because while our training set is substantial, it is still a relatively small sampling compared to the larger context in which thousands of games are played in the NBA in a single year. The ROC curve suggests the upper limit for a model's performance given access to a larger training set [24].

***Figure** 8: Metrics for Classification Models*

5.3.2 Deep Learning Models

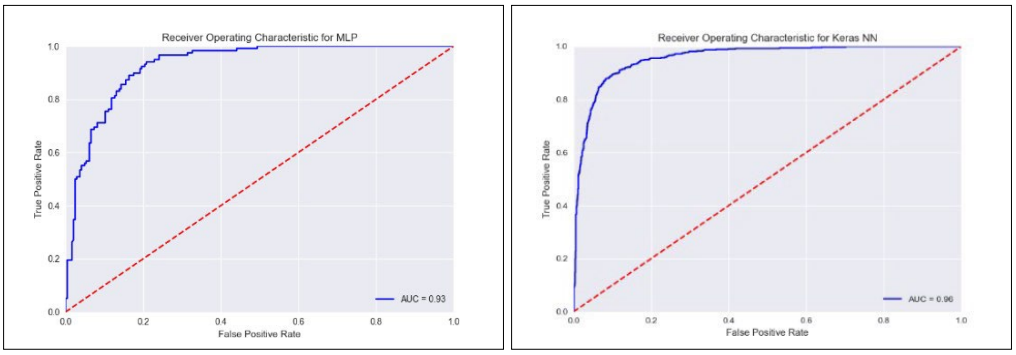
In addition to the three supervised learning approaches discussed in the previous section, this paper implemented two neural networks to evaluate the effectiveness of deep learning on the DHO candidate dataset. As seen in Figure 9, neither the Scikit-learn MLP nor the Keras/TensorFlow algorithms significantly outperformed SVM in terms of accuracy, precision, nor recall.

***Figure** 9: Metrics for Deep Learning Models*



Both neural networks, however, scored significantly higher on AUC, each topping 90% as seen in Figure 10. This makes sense, as the appeal to deep learning is that the models themselves decide how to weight and value features, often identifying and capitalizing on patterns that are difficult for human researchers to deduce after the fact.

***Figure** 10: ROC Curves for Deep Learning Models*



5.3.3 Clustering Analysis

Finally, ten clusters were formed from the 570 images. Many of these clusters exhibit strong similarities and distinguishing characteristics, whether that be the location on the court where the action occurred, the distance the ball traveled during the pass, or the speed and angle at which the screening and cutting players moved. While many of the clusters have a similar population, there is a significant range between the maximum and minimum populations, which is not uncharacteristic for a live dataset containing a large number of variants.

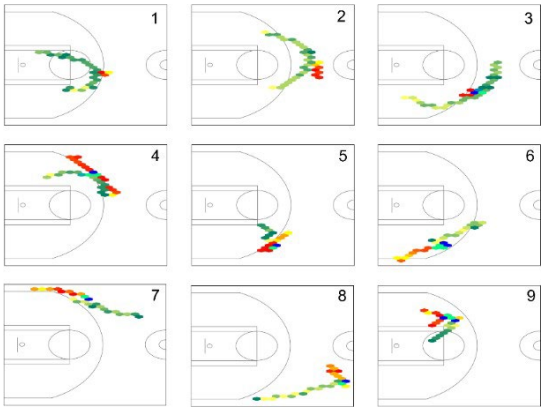
***Table** 5. Overview of Cluster Size and Distribution*

Cluster Overview					
Cluster	***# Samples***	***Total Distortion***	***Cluster***	***# Samples***	***Total Distortion***
1	123	79,368,671,504	6	146	87,575,033,771
2	137	92,830,919,208	7	116	73,878,921,094
3	74	48,534,571,052	8	99	64,499,359,866
4	141	84,456,640,099	9	144	87,264,699,916
5	113	66,404,011,454	-	-	-

Considering the vast amount of data collected each season in the NBA, the 43 games used for this analysis represent a small sample. As a result, some patterns that may be present in the dataset may not be well represented enough to merit their own clusters. Additionally, since not all actions go according to plan, some actions in the dataset may not have been performed well and may have even encountered some obstacle along the way that altered their execution entirely.

Figure 11 below shows the top representative samples from each cluster. Top samples are defined as those closest to the centroid of the cluster. This is defined using similarity scores to evaluate distances, and samples closer to the center of a cluster can be said to better represent the pattern present within. Overall, the similarities in court location, pass distance, and player trajectory are encouraging signs that the clustering algorithm is correctly identifying basketball strategies within the actions.

***Figure** 11: Example Action Hex Maps from Resulting Clusters*

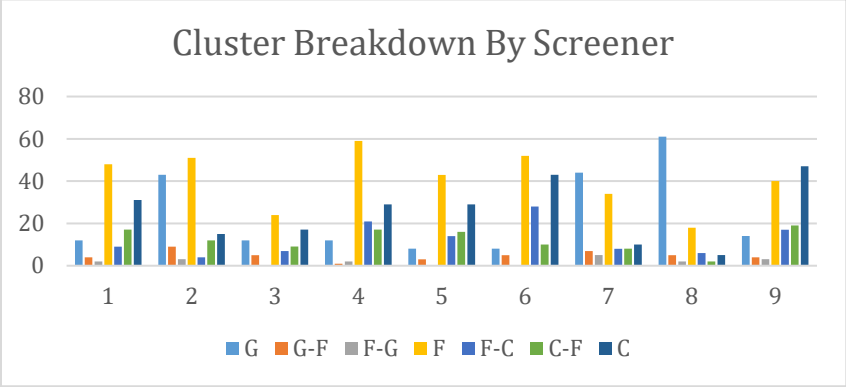


5.3.3.1 Cluster Population by Player Archetype

Once the clusters have been formed, a variety of metrics and snapshots can be generated to examine the trends within the emerging patterns. Previous clustering efforts in literature have examined how a team's strategy evolves and shifts from year to year, or even quarter to quarter. Others have looked at the distribution of actions across different players, which indicate different roles and play styles among and between player roles.

In Figure 12 below, we break the clusters down by the screener's archetype. While DHOs are often initiated by larger, less mobile players as a way to create space for quicker, more dynamic players, this isn't always the case. As we can see below, most clusters tend to have a higher distribution of frontcourt players (F-C, C-F, C), though some notably do not. For example, cluster 8 is primarily guards. This makes sense, as a frequent offensive initiator is a quick DHO involving the player bringing the ball up the court. Similarly, some clusters notably contain a great deal more forwards (or wing players), which likely indicates a strategic difference in execution. While not a hard rule, wings tend to be better perimeter shooters than frontcourt players, so it makes sense that forward-dominated clusters, such as cluster 7, tend to be closer to the three-point line, with the screener more likely to flare out for a shot than roll to the basket for a layup attempt. It should be noted that while the archetype breakdown is informed from the source data, the relative populations of certain roles (such as G-F, F-G) are significantly less populated than others (such as G, F), which contribute to the skew of the plot.

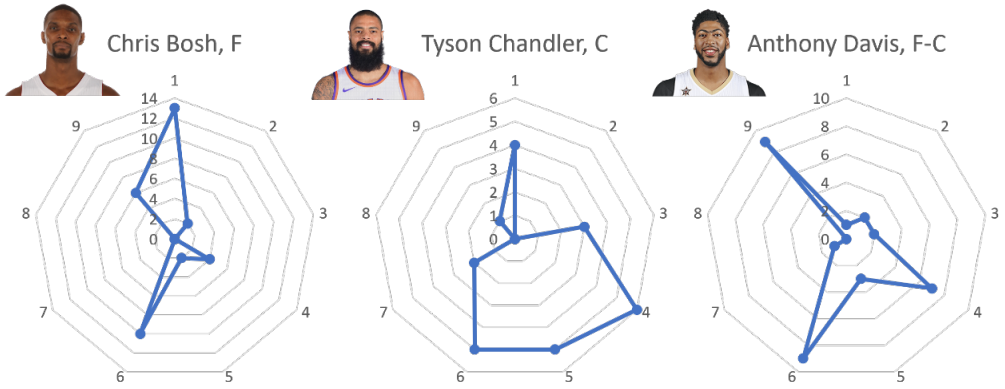
***Figure** 12: Cluster Breakdown by Screener Archetype*



5.3.3.2 Player Profiles

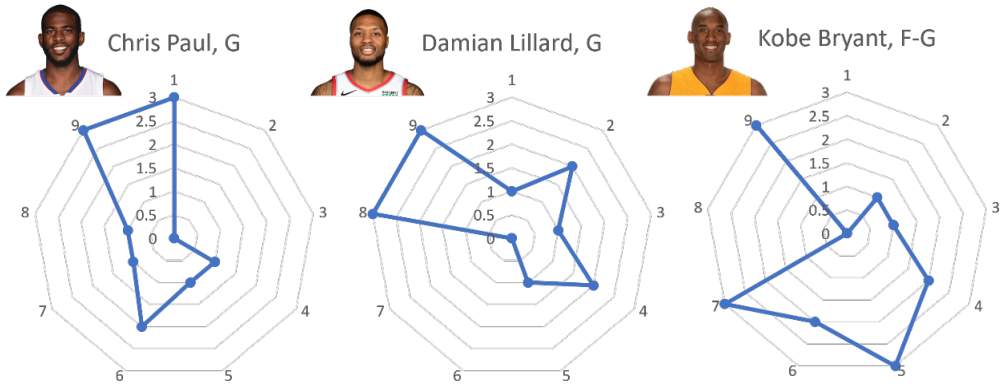
Perhaps the most interesting insights the action clusters can provide are into the tendencies of particular players. In Figure 13 below, we have radar charts indicating the frequency of involvement in each cluster by three screening players. Interestingly, each profile is rather distinct and can help point to the differences in players' skillsets. Here, Bosh is identified as a forward, and as expected, shares more in common with the hybrid forward-center Davis. Chandler, by comparison, is a traditional center without the diverse offensive skillsets of the other two players, and his profile differs more significantly from the other two, indicating a different form of offensive utilization.

***Figure** 13: Cluster Profiles for Screening Players*



In Figure 14, we examine the difference between cutting players. Similar to the screening roles, player archetype makes a difference here, as demonstrated by the hybrid forward-guard Bryant. Conversely, Paul and Lillard are both point guards, but as the variance in cluster membership indicates, their offensive roles are very different. Paul is considered a "traditional point-guard", who tends to prioritize passing and playmaking. Lillard, however, is an example of a more modern "score-first point-guard", who still maintains primary playmaking responsibilities but whose central offensive role entails a greater degree of shooting and scoring. This is an interesting way in which action variance can be a better indicator of a player's contributions to the team than traditional archetypes and roles.

***Figure** 14: Cluster Profiles for Cutting Players*



6. Conclusion and Future Work

This paper proposes an end-to-end approach to automatically detect DHO actions. First, it surveys approaches to NBA content analysis, the types of machine learning algorithms used to classify and extract patterns from player movement data, the challenges with feature engineering and noise filtration present in these pipelines, and the current solutions implemented in this domain. This paper presents a constructed pipeline capable of ingesting raw data and producing a classified dataset of DHO actions in a number of approach and execution strategies, including fakes. It assembles a large manually-labeled dataset for training, explores new approaches in data representation and feature engineering within NBA action classification, and presents three supervised and two deep learning models for evaluation. Hex map representations of the actions are then clustered in an unsupervised, bottom-up approach to pattern extraction that can be used to evaluate strategic differences in action variants as well as the skillsets of the players performing them.

In future work, the results of these action clusters can be taken into consideration along with the spatial alignment of players and the concept of court realty to provide a contextual and non-outcome-based quality metric to compare actions [26]. Additionally, clustered player tendencies could be introduced as an alternative to traditional player archetypes [18]. The movements of defensive players and their strategies in defending DHOs could be plotted and explored in a similar manner, as they have been for on-ball screens [3]. Other on- or off-ball actions could be added via new rules sets for the candidate algorithm and feature vectors for the classifiers to study different player movements. A natural language layer could be superimposed upon the final dataset to allow for high-level querying and meta-analysis [10]. As both basketball and machine learning strategies continue to develop, there will continue to be novel ways to interpret data, represent meaning, and train learning models. While the explicit contribution of this work implements an end-to-end classification pipeline, the hope is that it provides a foundation for further action detection classification, pattern mining, and analytics research and that it inspires others in the domain to think creatively about how to embed the semantics of basketball into machine learning applications.

ACKNOWLEDGMENT

This work was supported in part by the Niswonger Research Fellowship in Computer Science. The authors would like to thank the Niswonger Foundation for the support of this research.

References

[1] A. Nistala and J. Guttag, "Using Deep Learning to Understand Patterns of Player Movement in the NBA," in *Proceedings of the MIT Sloan Sports Analytics Conference*, 2019, pp. 1–14.

[2] "STATS SportVU®Basketball Player Tracking." .

[3] A. McIntyre, J. Brooks, J. Guttag, and J. Wiens, "Recognizing and Analyzing Ball Screen Defense in the NBA," in *Proceedings of the MIT Sloan Sports Analytics Conference, Boston, MA, USA*, 2016, pp. 11–12.

[4] A. McQueen, J. Wiens, and J. Guttag, "Automatically Recognizing On-ball Screens," 2014.

[5] N. Seward, "Building an On-ball Screen Dataset Using Supervised and Unsupervised Learning," University of Ontario Institute of Technology, 2018.

[6] J. D. Brooks, "Using Machine Learning to Derive Insights from Sports Location Data," Massachusetts Institute of Technology, 2018.

[7] S. Narayan, "Applications of Machine Learning: Basketball Strategy," Massachusetts Institute of Technology, 2019.

[8] D. K. Stephanos, G. Husari, B. T. Bennett, and E. Stephanos, "Machine Learning Predictive Analytics for Player Movement Prediction in NBA: Applications, Opportunities, and Challenges," in *Proceedings of the 2021 ACM Southeast Conference*, 2021, pp. 2–8, doi: 10.1145/3409334.3452064.

[9] D. H. Fisher, M. J. Pazzani, and P. Langley, *Concept Formation: Knowledge and Experience in Unsupervised Learning*. Morgan Kaufmann, 2014.

[10] M. Kates, "Player Motion Analysis: Automatically Classifying NBA Plays," Massachusetts Institute of Technology, 2014.

[11] A. Yu and S. S. Chung, "Automatic Identification and Analysis of Basketball Plays: NBA On-Ball Screens," in *2019 IEEE International Conference on Big Data, Cloud Computing, Data Science \& Engineering (BCD)*, 2019, pp. 29–34.

[12] K.-C. Wang and R. Zemel, "Classifying NBA Offensive Plays Using Neural Networks," in *Proceedings of MIT Sloan Sports Analytics Conference*, 2016, vol. 4.

[13] A. Nistala, "Using deep learning to understand patterns of player movement in basketball," 2018.

[14] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[15] D. Lutz, "A Cluster Analysis of NBA Players," in *Proceedings of the MIT Sloan Sports Analytics Conference, Boston, MA, USA.*, 2012, vol. 24, p. 2016.

[16] A. Franks, A. Miller, L. Bornn, K. Goldsberry, and others, "Characterizing the Spatial Structure of Defensive Skill in Professional Basketball," *Ann. Appl. Stat.*, vol. 9, no. 1, pp. 94–121, 2015.

[17] J. Sampaio, T. McGarry, J. Calleja-González, S. Jiménez Sáiz, X. i del Alcázar, and M. Balciunas, "Exploring Game Performance in the National Basketball Association Using Player Tracking Data," *PLoS One*, vol. 10, no. 7, p. e0132894, 2015.

[18] S. Kalman and J. Bosch, "NBA Lineup Analysis on Clustered Player Tendencies: A new approach to the positions of basketball \& modeling lineup efficiency of soft lineup

aggregates," 2020.
[19] J. Sampaio, E. J. Drinkwater, and N. M. Leite, "Effects of Season Period, Team Quality, and Playing Time on Basketball Players' Game-related Statistics," *Eur. J. Sport Sci.*, vol. 10, no. 2, pp. 141–149, 2010.
[20] M. Teramoto, C. L. Cross, R. H. Rieger, T. G. Maak, and S. E. Willick, "Predictive Validity of National Basketball Association Draft Combine on Future Performance," *J. Strength \& Cond. Res.*, vol. 32, no. 2, pp. 396–408, 2018.
[21] K. Saini, Udam; Evans, "No Title," 2017. https://eightthirtyfour.com/data (accessed Jan. 09, 2020).
[22] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009, doi: 10.1145/1656274.1656278.
[23] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Incorporated, 2019.
[24] C. E. Metz, "Basic principles of ROC analysis," *Semin. Nucl. Med.*, vol. 8, no. 4, pp. 283–298, 1978, doi: 10.1016/S0001-2998(78)80014-2.
[25] B. Azhagusundari and A. S. Thanamani, "Feature Selection based on Information Gain," *Int. J. Innov. Technol. Explor. Eng.*, no. 2, p. 18, 2013, doi: 10.1016/j.asoc.2008.05.006.
[26] L. B. Dan Cervone and K. Goldsberry, "NBA Court Realty," 2016.