Classification of Common Basketball Actions Using Player Tracking Data from the ShotTracker System

by

Scott Baker

B.S., University of Colorado, 2020

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirements for the degree of
Master of Science
Department of Applied Mathematics
2020

Committee Members:
Eric Vance
Brian Zaharatos
Anne Dougherty

Baker, Scott (M.S., Applied Mathematics)

Classification of Common Basketball Actions Using Player Tracking Data from the ShotTracker System

Thesis directed by Prof. Eric Vance

ShotTracker provides a set of sensors and products for basketball teams to automate their collection of in-house data and statistics. In addition to common box-score statistics and heatmaps, this system provides three-dimensional location-based tracking data of players and balls at a rate of three Hertz. The primary goal of this research is to use the location-based tracking data in conjunction with common machine learning classification techniques such as artificial neural networks (ANNs), recurrent neural networks (RNNs), and support vector machines (SVMs) to classify seven common actions that happen in basketball: on-ball screen, pindown screen, flare screen, backdoor cut, UCLA screen, stagger screen, and handoff. The resulting average testing set accuracy for using ANNs and RNNs is 90% while using SVMs is 75%. In addition to discussing accuracy results, two applications of these classifications are presented. First, a semi-supervised approach is presented based on these classifiers to label actions in additional possessions. Second, as an application to basketball, possession efficiency is analyzed and compared using the actions labeled within plays. Ultimately, creating reporting-based box-score statistics based on this action classification will allow teams to get an in-depth look into their possessions and can provide another level of usefulness to the statistics currently presented by the ShotTracker system.

Dedication

For Coach J, Coach T, Shan, Alex, Jill, and the Colorado Women's Basketball program.

Acknowledgements

Thank you to the following people who helped with this work:

Committee Members: Eric Vance, Brian Zaharatos, and Anne Dougherty

Jill Mahoney and Bill Cartun for approval to access the ShotTracker data.

JR Payne and Adam Ringler for spending time with me discussing how analytics are used from the perspective of a coach.

John Glenn and Corey Wacknov for help with the ShotTracker API.

Johnny Tamanaha for his advice on plotting the basketball courtlines.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

The main purpose of this research is to develop the ability to classify common actions that happen within a basketball possession using data from the ShotTracker system. After developing an understanding of classification, these classifications are applied to a team's possessions in order to understand their offensive efficiency in terms of Points Per Possession (PPP). In addition to comparing the offensive efficiency with labeled data, the offensive efficiency of labeled and unlabeled data is analyzed based on classifiers built from a semi-supervised learning approach. Breaking down basketball possessions into specific actions allows coaches to understand which actions result in better offensive outcomes. The purpose of this research is to use machine learning to automate the labeling of these action-based classifications from the location-based tracking data from the ShotTracker system. In practice, this work shows a proof-of-concept system of action-based analysis which could develop into a feature of the ShotTracker system that would allow coaches and players to see a a team-based breakdown of PPP by actions within their plays.

1.1 Motivation

With the rise of data being used in industries such as technology, health care, and geological sciences, it has also been introduced in sports. The book *Moneyball* [5] brought the use of analytics in baseball scouting to the mainstream media. The analysis of baseball statistics, called sabermetrics, then inspired other sports to follow suit. Basketball is a sport that has started to adopt the use of analytics for player evaluation and evolution of the game. Kirk Goldsberry, who emerged when advanced analytics started popping up in the National Basketball Association (NBA), mentions in his book *SprawlBall* [2] that ten years ago, analytics in the NBA was "all about spreadsheets and linear regression."

From this, Kirk went on to be the first person to do a widespread analysis of location-based shot data. This analysis, originally called CourtVision and later developed into Spread, made heatmaps of a player or team's shot percentage based on the location the shot was taken on the floor. These heatmaps were used to analyze efficiencies of a player or team. Daryl Morey, the general manager of the Houston Rockets, used similar information to figure out that based on current trends in shooting percentage, a three-point shot is more valuable per possession than a long two-point shot [2]. This led to a shift in the types of players that they sought out—specifically players that could shoot the three.

Similar to Morey's analysis, this paper intends on developing some analytics for reporting purposes, rather than predictive ones. First, we use possession-based movement data from the ShotTracker [10] system to train a series of supervised classifiers to identify seven common basketball actions: ball screen, UCLA screen, pin-down screen, flare screen, stagger screen, backdoor cut, and rub-cut/handoff. After training supervised classifiers, a semi-supervised learning technique, transfer learning, is tested in order to validate the accuracy of labeling new data. In addition, we analyze the relationship between the actions that happen on a given play and the outcome of the possession in terms of a statistic, Points Per Possession (PPP).

In this paper, we present the format of the location-based tracking data and the data engineering techniques in Section 2. After preparing and labeling the data, classification techniques such as Artificial Neural Networks (ANNs), Recurrent Neural Networks (RNNs) and Support Vector Machines (SVMs) are discussed in Section 3. Then, an analysis of possession efficiency and an exploration into labeling more data for further possession efficiency analysis is presented in Section 4. In Section 5, this thesis is concluded with a discussion of future work.

1.2 Philosophy and approach

Goldsberry's work to chart out every shot is a valuable analysis into the state of the game. There are three main purposes for statistics: exploration, inference, and explanation. From our approach, the use of advanced analytics such as Spread should be oriented towards exploration and explanation. This is because decisions about split-second events in a game come from the experience of the coaches and players. For example, even though predictions can be made about a shot from a particular player for a point in the future, the usefulness of this information is low. There is no reason to predict whether or not Steph Curry will shoot another three-pointer from 29 feet in front of the opposing bench sometime in the next game. However, retrospectively seeing the heatmap of Curry's shots could influence how he focuses his workouts to practice shots from a lower-percentage area.

As a step back, inference in analytics should focus on *how can that player improve their game with advanced analytics?* rather than *where should that player take their next shot?*

As fans, players, coaches, officials, and data scientists, there is an underlying beauty to some of the randomness in sports, especially in basketball. If this randomness can be completely analyzed and charted out, then the beauty of the game starts to diminish. The advantage to classifying actions within plays and reporting the relationship to the outcome of plays is that it will allow coaches to develop and improve their offenses based on the strengths of their team. This type of reporting is not oriented towards predicting when and where a team will set their next ball screen. As Tobias Moskowitz and L. Jon Wertheim mention in their book *Scorecasting: The Hidden Influences Behind How Sports Are Played and Games Are Won* [18], "[your team] may have a better strategy, but if the athletes don't buy in, it's probably not worth deploying."

Without player buy-in and the coach-player relationship, it can be difficult to have success as a team. The "smallball" approach of the Houston Rockets would not be as effective if Mike D'Antoni, who is an offensive-minded coach, did not get his players to buy into the "run-and-gun" offensive tactics based on spreading the floor and focusing on the three-point shot. Further, acquiring players such as James Harden and Russell Westbrook, elite isolation players who know how to pass [2], and surrounding them with shooters such as Austin Rivers, Eric Gordon, Ben McLemore, and others, would allow the Houston Rockets to take advantage of the advanced shot-based efficiency analysis brought to light by Darryl Morey. Instead of changing the previous team they had based on these findings, Morey overhauled the previous roster and staff in order to make room for this current group of players to take advantage of the analytics.

The point of using analytics for reporting purposes is that the cohesiveness of a team is built on the relationships within the team. Further, the role of analytics is to help the team develop. The intent of creating reporting-based statitstcs based on action classification with the ShotTracker system is that the efficiencies can be leveraged by coaches with access to this technology to design practices and workouts with the goal of helping their players improve and their team win more games.

Chapter 2

Data and The ShotTracker System

Before looking into action classification, some steps were taken to prepare the data. To begin, we discuss the ShotTracker system and features used to access the data. Once pulling the data, the techniques used to process this data are described. These techniques include filling in gaps, finding the correct players, and standardizing the data into a universal half-court axis.

2.1 ShotTracker

During summer 2019, the Colorado Basketball program invested in a ShotTracker system for all three of their gymnasiums. The addition of this system helps coaches automate the tracking of the performance of their players. Before the use of ShotTracker, coaches and managers would have to manually input statistics if they wanted to track player performance, specifically in practice.

ShotTracker is a Kansas City-based company that was founded in 2013 by Bruce Ianni and Davyeon Ross [11]. The ShotTracker system automatically charts out over 70 statistics [9] for players and coaches to see in real-time or retrospectively through a mobile application or website. The system relies on a set of sensors placed on the gym walls and ceilings, a small matchbook-sized sensor that players attach to their shoes or jersey, and basketballs with a sensor in the center. While the system is in use, ShotTracker takes the location of all player-worn and ball sensors at a rate of 3 Hertz (Hz).

There are two direct benefits of having the ShotTracker system. First, players are able to see their statistics on the ShotTracker mobile application or website. This allows them to see how well they are shooting and playing on a daily, weekly, or even season-long basis. Second, coaches are able to see both individual and team performance. Similar to the view that the players get on their ShotTracker application, coaches can see heatmaps and shooting percentages broken down for individual players, a subset of the whole team, or the whole team on a weekly, monthly, or season-long basis. Additionally, the coaches' application allows them to compare players' statistics, test different lineups, and view full scrimmage or game possession data.

2.1.1 Dimensions of the court

A standard National Collegiate Athletic Association (NCAA) basketball court is 94 feet long and 50 feet wide. Figure 2.1 shows the layout of the court.

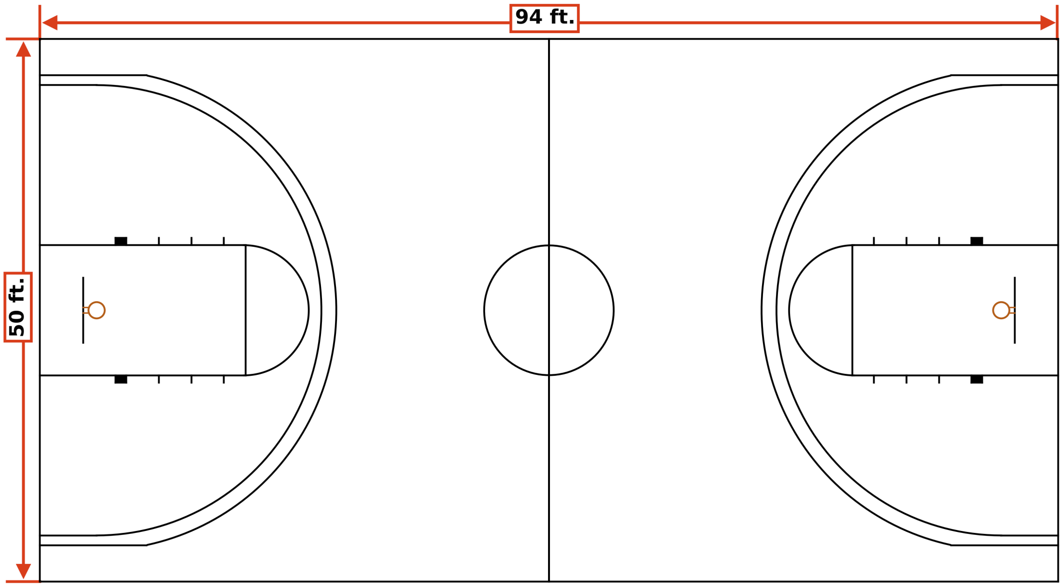

Figure 2.1: Dimensions of an NCAA basketball court

As of 2019, there is one difference in the courtlines for the Women's game and the Men's game. The Women's game has a three-point line that is 20 feet and 9 inches away from the hoop whereas the Men's game has a three-point line 22 feet and 1.75 inches away from the hoop [19]. Although these differences might affect overall scoring, it does not effect the types and locations of actions happening within possessions.

The sensors in the basketball and worn by the players record three-dimensional locations to millimeter-level accuracy at 3 Hz. This means that for location-based data, the coordinate system set up by ShotTracker for use on their Application Programming Interface (API) has units of millimeters. Figure 2.2 shows the coordinate system set internally by ShotTracker for data recording.

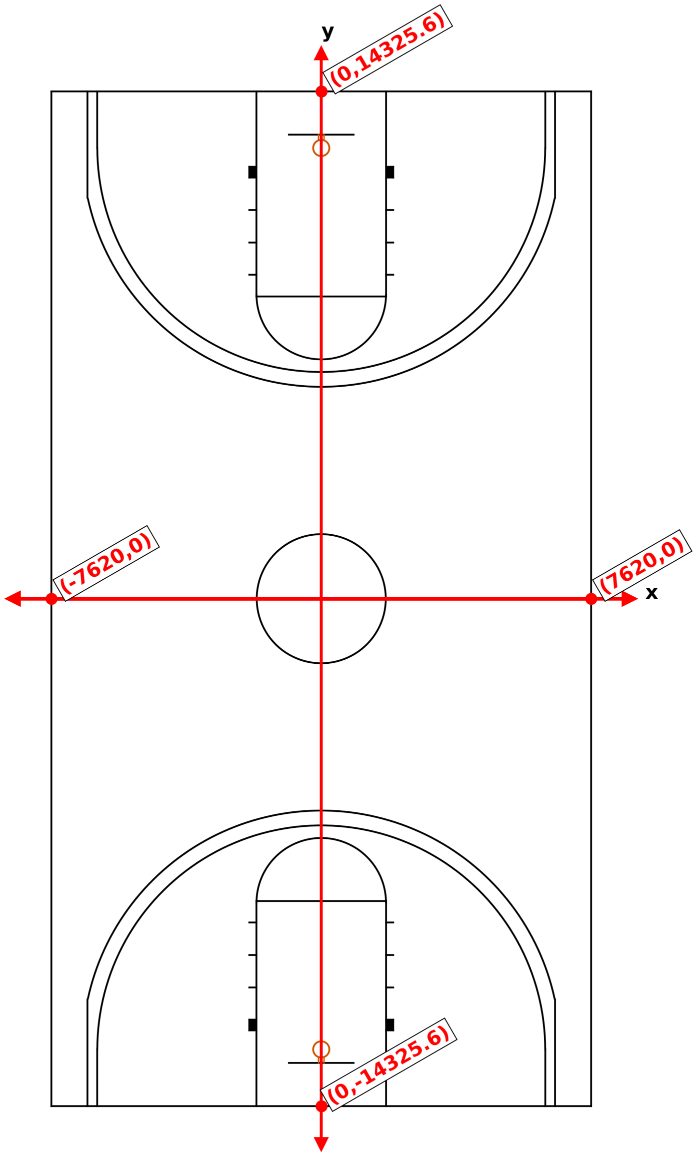

Figure 2.2: Standard (x, y) coordinate system for location based data from ShotTracker system

This millimeter-based coordinate system is standard across all courts that are tracked with ShotTracker. When the system is first installed at a location, ShotTracker calibrates each court to this standard axis. Specifically, the point $(0, 0, 0)$ is on the floor at center court. As we will explain later, the z-coordinate is not used since it is not significant for classification. This means that for all three gyms at the CU Events Center, information from the sensors will all map to the same axis. Additionally, when referring to the ends of the court for the transformation of coordinates in Section 2.2.5, the “north” end is where $y \geq 0$ and the “south” end is where $y < 0$.

2.1.2 Using the ShotTracker coordinate system

In addition to this standardized axis mapping between courts, ShotTracker implements a hoop-based coordinate system. The hoop-based coordinate system translates any shot on any hoop in a ShotTracker-configured gym onto an axis that has the point $(0, 0, 0)$ on the floor underneath the center of the hoop. The benefit of having a shot-centered coordinate system is that ShotTracker can internally make heatmaps resembling those made by Goldsberry from shots on any hoop in the gym.

A similar coordinate system mapping is built for this paper, which maps all possession data from either end of the court onto one side of the court. The benefit of this full-court to half-court mapping is to create an environment that accurately represents possessions on either end of the court. This works because the plays and actions within basketball should be the same, regardless of the end of the court they happen on or the gym they happen in. Details of our approach to a standard coordinate system are discussed in Section 2.2.5.

2.2 The data

In addition to accessing the data through the team application, the raw data can also be accessed through ShotTracker’s Application Programming Interface (API) [12]. For the purpose of this paper, location-based possession data and associated box-score statistics are accessed through this API. To satisfy NCAA compliance, all data in this paper has been de-identified because of the author’s relationship with the Colorado Basketball program. However, this paper serves as a proof-of-concept for these reporting-based advanced analytics. Therefore, for the purpose of this manuscript, all specific data-related results and statistics will be associated with a hypothetical team, the BlueJays.

2.2.1 An overview of possessions

In total, there are 6236 possessions available for analysis. These possessions come from three different scenarios: drills, 5-on-5 scrimmages, and games.

Since the ShotTracker system is always recording during these drills, scrimmages, and games, a couple of complications arise when trying to look through the possession data. These issues and our data engineering solutions are addressed in Section 2.2.4.

2.2.2 Format of a location placement

To build up the discussion of our data engineering process, the most simple element of data required is the position of a sensor at a certain time. Through the API, this position is returned in a JavaScript Object Notation (JSON) format. A particular example of this location is in Figure 2.3.

```
"locations": [
    {
      "at": 1577836800000,
      "placements": [
        {
          "id": 01738,
          "type": "PLAYER",
          "x": 992,
          "y": -12001,
          "z": 511
        }
        ...
        ]
    }
]
```

Figure 2.3: Sample output for API location request to sensor data within a possession

There are four attributes to a request of the `"locations"` data. First, there is a timestamp labeled `"at"` which is the time of recording, in Epoch time. This UNIX Epoch time represents the number of milliseconds that have passed since January 1, 1970. For each Epoch timestamp, there are `"placements"` for each of the players in the gym and each ball in use. The second attribute to the location is the `"id"` found inside of the `"placements"`, which is the player's unique identification. Third, each sensor has a `"type": "BALL"` or `"PLAYER"`. Distinguishing the type of sensor recorded will be important in Section 3.1, since only some common basketball actions depend on the ball being present. Finally, each placement has a (x, y, z) coordinate location in the gym on the full-court ShotTracker axis.

An important note is that the z-coordinate for these locations, which represents the height of the sensor above the surface of the court, was not used as an input to the classification models. Excluding the z-coordinate comes from the potential extra error introduced from mixing the multiple types of player sensors that ShotTracker offers. One type of sensor available attaches to a player's shoelaces and another type attaches to a player's jersey. In general, the average height of a player's jersey is higher than the average height of a player's shoe. Since the z-coordinate represents the height of a sensor from the floor and both of these sensor types are used, including the z-coordinate would add unnecessary noise to the data. Having a mix of sensor type, which is positioned on different parts of a player, also means that extra noise could be added to both the x-coordinate and the y-coordinate. This potential source of error is beyond the scope of this paper and will be investigated in future development of this research.

2.2.3 Format of the possession data

Within a possession on the API, there are two main objects that are used. First, there are a series of `"locations"`, which has the structure shown in Figure 2.3. Second, as an application in Section 4.2, `"stats"` are taken for each possession. Possibilities for these statistics include field goal (`"FG"`), field goal attempt (`"FGA"`), assist (`"AST"`), offensive rebound (`"OFFENSIVE_RB"`), defensive rebound (`"DEFENSIVE_RB"`), and turnover (`"TO"`) [CITE]. An example of the recorded possession statistics output is in Figure 2.4.

```
"stats": [
    {
        "timestamp": 1577836800000,
        "possession_type": "FG",
        "player_id": 01738,
        "shot_attributes": {
            "shot_distance_cm": 719.6249370331742,
            "x": 7187,
            "y": 13236,
            "z": -1,
            "is_3point": true
        }
    }
...
]
```

Figure 2.4: An example of the JSON output associated with accessing possession statistics through the API

2.2.4 Data processing and engineering

For processing the possession data, the first step is to take all positions at a timestamp and create a table such as the one in Figure 2.5.

```
index          id    type     x      y     z
8      2140000000    BALL  -103 -12944  3221
1           01738  PLAYER  3473  -7150   124
10          05679  PLAYER -5553 -12234    42
9           01730  PLAYER  -636 -11025   668
7           01737  PLAYER  -314 -14051    39
6           01731  PLAYER   357 -11243   254
2           01734  PLAYER  1374 -12374     1
5           05680  PLAYER  2487 -13149    37
3           05280  PLAYER   833 -12382     6
4           05327  PLAYER -1690 -10354   706
0           05345  PLAYER -1013 -12652    45
```

Figure 2.5: Output table created for each timestamp showing the locations of all available sensors

This table is created for each timestamp, or frame, within the possession. For example, a possession that lasted 24 seconds would have about 72 frames. Sometimes at a particular timestamp, this table has more than ten players and more than a single ball. An example of this would look like the table shown in Figure 2.6.

```
index          id    type     x       y     z
8      2140000000    BALL  -103  -12944  3221
1           01739  PLAYER  3473   -7150   124
10          05679  PLAYER -5553  -12234    42
9           01730  PLAYER  -636  -11025   668
12          01736  PLAYER  7714   15001    39
7           01737  PLAYER  -314  -14051    39
6           01731  PLAYER   357  -11243   254
2           01734  PLAYER  1374  -12374     1
5           05680  PLAYER  2487  -13149    37
11     2150000000    BALL  -227  -15425    12
3           05280  PLAYER   833  -12382     6
4           05327  PLAYER -1690  -10354   706
0           05345  PLAYER -1013  -12652    45
```

Figure 2.6: Output table created for each timestamp showing the locations of all available sensors with more than one ball and ten players present in the frame

In the game possession data, it is clear which players are on the court and which team the players belong to. However, for the drill-based and scrimmage possession data, which are internal team events, the system fails to categorize which players are on which "teams" some of the time. This leads to two things happening. First, it fails to recognize possessions in which more than ten players are on the court sometime in the duration of the possession. An example of this is when there is a substitution right before the basketball possession starts but after the system starts recording data for the possession. The second scenario is when there are other players or balls in the gym but not in-bounds on the court.

Our data engineering solution takes care of both scenarios. First, at initial glance for the entire possession, the sensors that are off of the court for each timestamp are filtered out. The maximum and minimum values of being in-bounds on the ShotTracker court axis are seen in Table 2.1.

`coordinate`	`min`	`max`
x	-7620	7620
y	-14325.6	14325.6

Table 2.1: The minimum and maximum x and y coordinate values for players and balls that are in-bounds on a ShotTracker court

Next, looking over the entire possession for each player, the number of frames in which the player exists in-bounds are counted. Likewise, the number of frames in which the ball exists in-bounds are counted. These aggregates are sorted and used to decide which players and ball to include in the possession. The top-ranking ball is used and the top-ten players are used. Within a 46 frame possession, an example of this aggregation is seen in Figure 2.7.

```
rank       ball_id  frame_count          rank    player_id  frame_count
   1    2140000000           46             1        01739           46
   2    2150000000            8             2        05679           46
   3    2160000000            8             3        01730           46
                                            4        01731           46
                                            5        01734           46
                                            6        05680           46
                                            7        05280           46
                                            8        05327           46
                                            9        05345           45
                                           10        01736           45
                                           11        01717           12
                                           12        01718            4
                                           13        01779            1
```

Figure 2.7: An example of ranking aggregate ball (left) and player (right) frame counts for a singular possession

In this example, it appears that there is a ball that exists in-bounds for all of the 46 frames and there are ten players that are present in at least 45 of the 46 frames.

Sometimes, for technical reasons, the ShotTracker system does not record the locations of some sensors at some points. After selecting the most visible players for the possession, the next step is to fill in any potentially missing data. This is important because filling in holes in the data will simulate the natural movement of players. Of the possessions used for classification, 6% of all location data points are missing. Sensitivity analysis of filling in data is beyond the scope of this paper but will be investigated in future research. These holes in the data can be fixed in three ways.

In what follows, let $\left(x_t^{(i)}, y_t^{(i)}\right)$ denote the position of sensor i at time t. For a general possession, let $t = 0, 1, 2, ..., T$ denote the frame number within the possession and $i = 0, 1, 2..., N$ denote the number of sensors in the possession. For the most common case of a possession after the ranking-based filtering, $N = 10$. This means there are 11 sensors on the court, one for the ball and 10 for the players.

To look at these cases, assume we are interested in a particular player or ball, $i = k$. For each possession, this processing happens for all sensors in the possession.

The first case to consider is when there is a missing value for player k at $t = 0$, which is the first frame in the sequence. To handle this, the point $\left(x_0^{(k)}, y_0^{(k)}\right)$ is filled with a permutation of the next frame, $\left(x_1^{(k)}, y_1^{(k)}\right)$ such that

$$\left(x_0^{(k)}, y_0^{(k)}\right) = \left(x_1^{(k)}, y_1^{(k)}\right) + \vec{\epsilon}; \qquad \vec{\epsilon} \sim \mathcal{N}(\vec{\mu}, \mathbf{\Sigma}) \tag{2.1}$$

where

$$\vec{\mu} = 0; \qquad \mathbf{\Sigma} = diag(\hat{\sigma}_x^2, \hat{\sigma}_y^2). \tag{2.2}$$

Next, this type of permutation is done similarly for a missing value at the end of the possession. For sensor k, if the last frame, $\left(x_T^{(k)}, y_T^{(k)}\right)$, is missing, it is filled with

$$\left(x_T^{(k)}, y_T^{(k)}\right) = \left(x_{T-1}^{(k)}, y_{T-1}^{(k)}\right) + \vec{\epsilon}; \qquad \vec{\epsilon} \sim \mathcal{N}(\vec{\mu}, \mathbf{\Sigma}) \tag{2.3}$$

where

$$\vec{\mu} = 0; \qquad \mathbf{\Sigma} = diag(\hat{\sigma}_x^2, \hat{\sigma}_y^2). \tag{2.4}$$

In both of these cases, $\hat{\sigma}_x^2$ and $\hat{\sigma}_y^2$ denote the variance in the movement of player k over the remaining frames they are present in. Although in terms of basketball, there could be more accurate methods of filling holes on the ends of the possession, generally actions that this paper aims to classify do not happen on the first or last frames of a possession. For example, it is not possible for Klay Thompson to receive a full pin-down screen in one-third of a second at the beginning of a possession.

Additionally, if there are multiple missing frames at the end of the possession for player k, this process is recursively performed, building from the last frame with available $\left(x_t^{(k)}, y_t^{(k)}\right)$ values. Likewise, this recursive strategy is used to fill in multiple frames in the beginning of a possession, building from the first frame with available $\left(x_t^{(k)}, y_t^{(k)}\right)$ values.

To fill in missing frames for player k from a frame in the middle of the possession, the approach uses a midpoint calculation with some added noise. For simplicity, assume that information is available for sensor k at all times except time $t = j$, $j = 1, 2, ..., T-1$. This means that $\left(x_{j-1}^{(k)}, y_{j-1}^{(k)}\right)$ and $\left(x_{j+1}^{(k)}, y_{j+1}^{(k)}\right)$ are non-empty. In this case, the position at $\left(x_j^{(k)}, y_j^{(k)}\right)$ is calculated by

$$\left(x_j^{(k)}, y_j^{(k)}\right) = \frac{\left[\left(x_{j-1}^{(k)}, y_{j-1}^{(k)}\right) + \left(x_{j+1}^{(k)}, y_{j+1}^{(k)}\right)\right]}{2} + \vec{\epsilon}; \qquad \vec{\epsilon} \sim \mathcal{N}(\vec{\mu}, \mathbf{\Sigma}) \tag{2.5}$$

where

$$\vec{\mu} = 0; \qquad \mathbf{\Sigma} = diag(\sigma_x^2, \sigma_y^2). \tag{2.6}$$

In this case, $\sigma_x^2 = 0.05$ and $\sigma_y^2 = 0.05$. With no basketball interpretation, noise is added to the data since the manufactured point comes from a linear combination of other existing points.

Without adding noise, no additional "natural" information is added to the sequence of frames. Therefore, adding this type of noise helps build more robust classifiers.

If there is a hole in the location data for the possession frames that is more than one frame wide, this midpoint-based calculation will generalize to multiple holes by recursively searching for the size of the hole and filling in the manufactured information to the necessary frames.

2.2.5 Transformation

Once all the holes in the possession frames are filled with manufactured data from Section 2.2.4, there is preprocessing necessary to prepare the data correctly for the classification models.

The first step in preprocessing is to transfer all possession data onto a standard half-court axis. Since the data can come from either half of the court and actions happen the same way on either end, all the locations are mapped to the standard half-court axis seen in Figure 2.8. To accomplish this, the average y-coordinate location for all the sensors determines what transformation is given to the data. The following two functions, $f(x_t^{(i)})$ and $g(y_t^{(i)})$ are used to describe the transformation.

$$f(x_t^{(i)}) = \begin{cases} -(x_t^{(i)}), & \frac{1}{NT}\sum_{i=0}^{N}\sum_{t=0}^{T} y_t^{(i)} \leq 0 \\ \\ x_t^{(i)}, & \frac{1}{NT}\sum_{i=0}^{N}\sum_{t=0}^{T} y_t^{(i)} > 0 \end{cases} \tag{2.7}$$

$$g(y_t^{(i)}) = |y_t^{(i)}| \tag{2.8}$$

The functions f and g are applied to all points $\left(x_t^{(i)}, y_t^{(i)}\right)$ in the possession frames. The quantity $\frac{1}{NT}\sum_{i=0}^{N}\sum_{t=0}^{T} y_t^{(i)}$ determines, by averaging the y-coordinate locations of all sensors in the frame, whether or not the possession happens on the "north" or "south" end of the court. This quantity indicator is accurate since an additional preprocessing step is taken to make sure the average y-coordinate location of the ball matches the average y-coordinate location of all the players in the possession frames. This means that any frames that happen "in transition" are trimmed so that the possession only happens within one side of the court. In terms of basketball, the common actions outlined in this paper generally do not happen "in transition".

Since the parsed possession only includes frames that exist with this average player-ball-location match, we can guarantee that the players are on one end of the court with $\frac{1}{NT}\sum_{i=0}^{N}\sum_{t=0}^{T} y_t^{(i)}$ determining if they are on the "north" or "south" half.

For each possession, once f and g have been applied to all positions $\left(x_t^{(i)}, y_t^{(i)}\right)$ in the frame, the data is shifted and standardized to reflect values of $x_t^{(i)}$ and $y_t^{(i)}$ such that:

$$-1 \leq x_t^{(i)} \leq 1 \tag{2.9}$$

$$-1 \leq y_t^{(i)} \leq 1 \tag{2.10}$$

Therefore, the standardized coordinate system for the possession data is reflected in Figure 2.8.

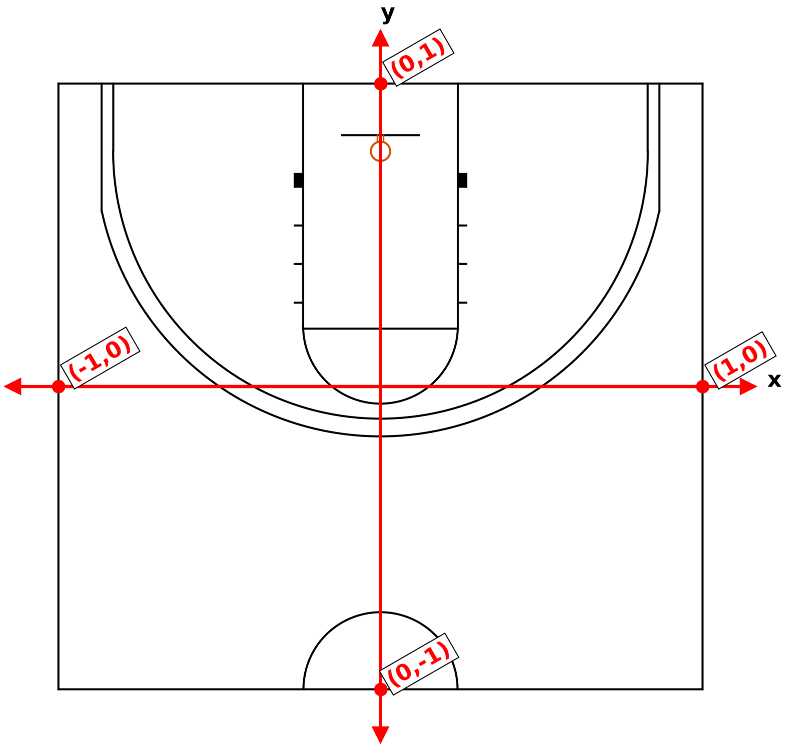

Figure 2.8: Final standardized and scaled half court axis for possession data

Standardizing these coordinates for the input data is an important step to help reflect the highest level of training accuracy for classification models.

2.2.6 Shapes of input

After standardization, the data is reshaped to reflect the appropriate inputs to the classification algorithms, which generally require a specific shape [15]. For the Vanilla Neural Networks (ANNs) described in Section 3.2.1, the input data is shaped as

$$\begin{pmatrix} y_0^{(0)} \\ x_0^{(0)} \\ y_0^{(1)} \\ x_0^{(1)} \\ \vdots \\ y_0^{(N)} \\ x_0^{(N)} \\ y_1^{(0)} \\ x_1^{(0)} \\ \vdots \\ \vdots \\ \vdots \\ y_T^{(N)} \\ x_T^{(N)} \end{pmatrix} \tag{2.11}$$

For the Recurrent Neural Networks (RNN) in Section 3.2.2, the shape of the input is

$$\begin{pmatrix} y_0^{(0)} & x_0^{(0)} & y_0^{(1)} & x_0^{(1)} & \dots & y_0^{(N)} & x_0^{(N)} \\ y_1^{(0)} & x_1^{(0)} & y_1^{(1)} & x_1^{(1)} & \dots & y_1^{(N)} & x_1^{(N)} \\ \vdots & & & & \ddots & & \vdots \\ y_T^{(0)} & x_T^{(0)} & y_T^{(1)} & x_T^{(1)} & \dots & y_T^{(N)} & x_T^{(N)} \end{pmatrix} \tag{2.12}$$

Since possessions have variable number of frames, input data reflects the appropriate level of padding [14] for the necessary classifier. Padding, by adding zeros for locations at the end of each possession, will allow possessions of different lengths to be processed in a these classification methods.

Chapter 3

Classification of Common Actions

The primary goal of this paper is to build, as a proof-of-concept, a series of machine learning classifiers that use the ShotTracker possession data to identify common actions that happen within a possession of basketball. Previous work has focused on trajectories [17], overall movement [16], and gaining information from videos [13]. With the use of the ShotTracker sensors, the aim of this paper is to build sensor-based classifiers, similar to ones that are found for the Human Activity Recognition dataset [1], that can identify the following actions: ball screen, pin-down screen, flare screen, stagger screen, UCLA screen, backdoor cut, and rub-cut or handoff. To begin, the types of basketball actions are discussed. This discussion is followed by a comparison of different types of machine learning algorithms for labeling the data, along with respective advantages and disadvantages.

3.1 Actions

Before discussing specific classification techniques, it is important to describe, in basketball terms, the seven common actions we wish to identify within a possession. These actions were chosen based on our basketball experience and the commonality of these actions for the Colorado Basketball teams. For better generalization, other actions that might be relevant to other teams will be investigated in future research. For each action, a sequence of frames is presented to visually show the action and is followed by a description. In the description, it is assumed that the defense is playing man-to-man and for every offensive player described, there is an associated defensive player guarding them.

An initial look at all of the labeled possessions (929) for the BlueJays revealed the commonality of each one of these actions, shown in Table 3.1.

ACTION	COUNT
BALL SCREEN	474
PIN-DOWN SCREEN	637
FLARE SCREEN	401
STAGGER SCREEN	46
UCLA SCREEN	73
BACKDOOR CUT	214
RUBCUT/HANDOFF	261

Table 3.1: Total counts from the labeled data of the seven common actions identified in a basketball possession

3.1.1 Ball Screen

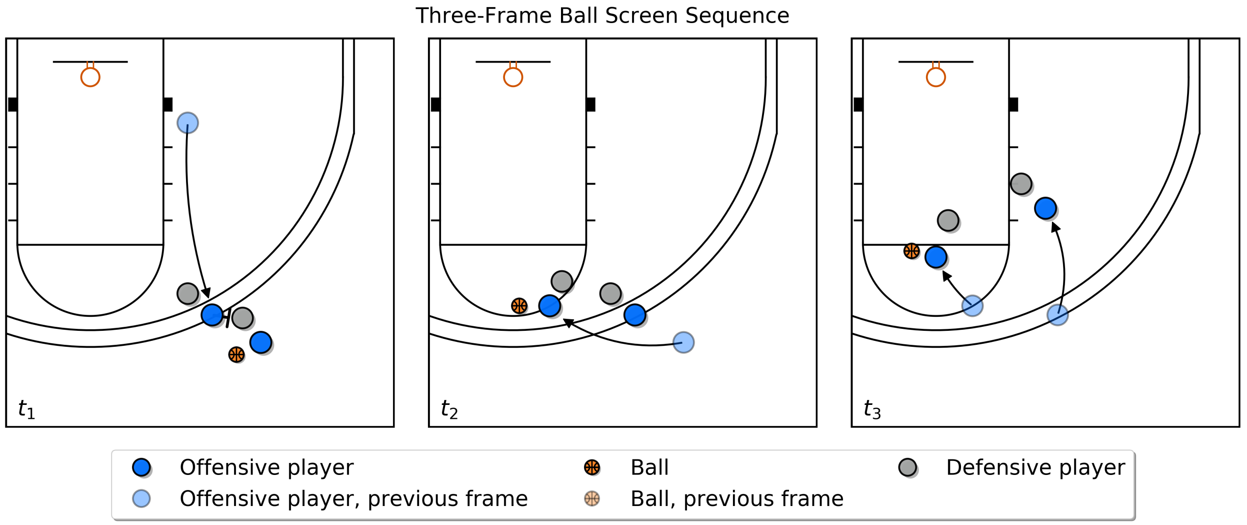

Figure 3.1: Three-frame sequence showing an example of a ball screen

A ball screen is an event where a screen is set by an offensive player on the defensive player guarding the ball-handler. In a single ball screen, there are two offensive players, one with the ball. Generally, this event happens around the three-point line at the top of the key or on either wing. The use of a ball screen in an offense is to trigger a choice-based response from the defense. When guarding a ball screen, the defensive players have to make the decision to either switch who they are guarding, hedge the screen (which means the defensive players attempt to make the ball-handler move back towards half-court), or shadow (letting the on-ball defender through the screen while the off-ball defender slows the ball-handler down). The variety of choices makes a ball screen a favorable action within plays. The offense has an advantage in ball screen scenarios since there are several options that come out of it and it is difficult for the defense to react and effectively guard all possibilities. These options are for the ball-handler to attack the basket, pass the ball to the player rolling to the basket after the screen, or kicking back to another player if their help-side defender gets involved in the action.

3.1.2 Pin-down Screen

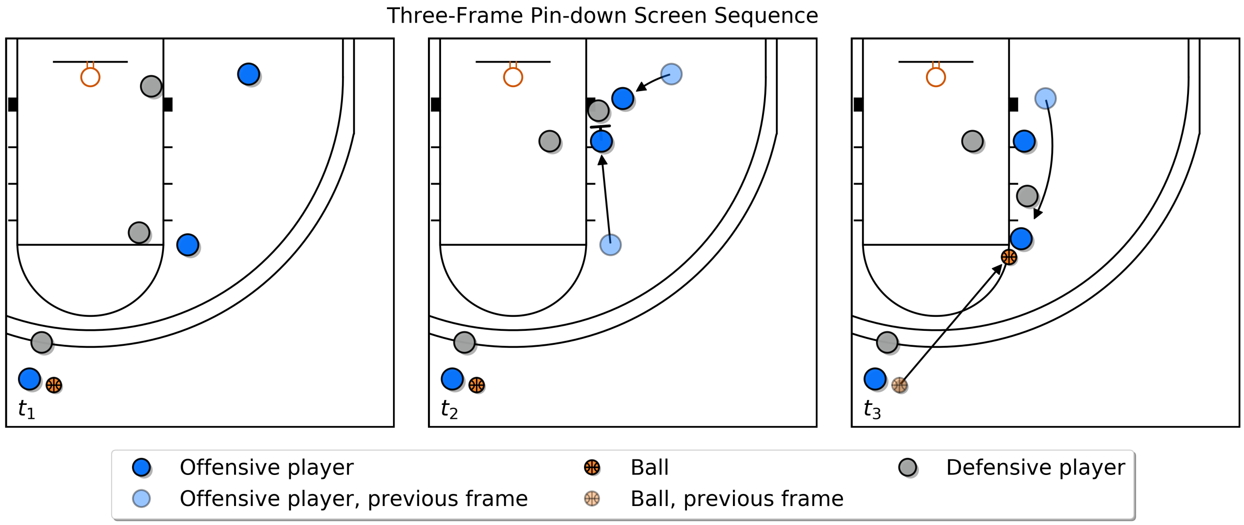

Figure 3.2: Three-frame sequence showing an example of a pin-down screen

A pin-down screen is an off-ball screen that happens with two offensive players. One of the offensive players sets a screen on the defender of the other offensive player. Pin-down screens are used for two reasons. First, they intend to create an opportunity for one offensive player to get open. Second, they allow the other offensive player to create position on their own defender for the opportunity to post-up or feed into another action. An important aspect to the pin-down screen is the direction that the screen is set. Usually the player doing the screening is pointed towards the basket, hence the “down” part of pin-down. These screens usually happen inside of the three-point arc.

3.1.3 Flare Screen

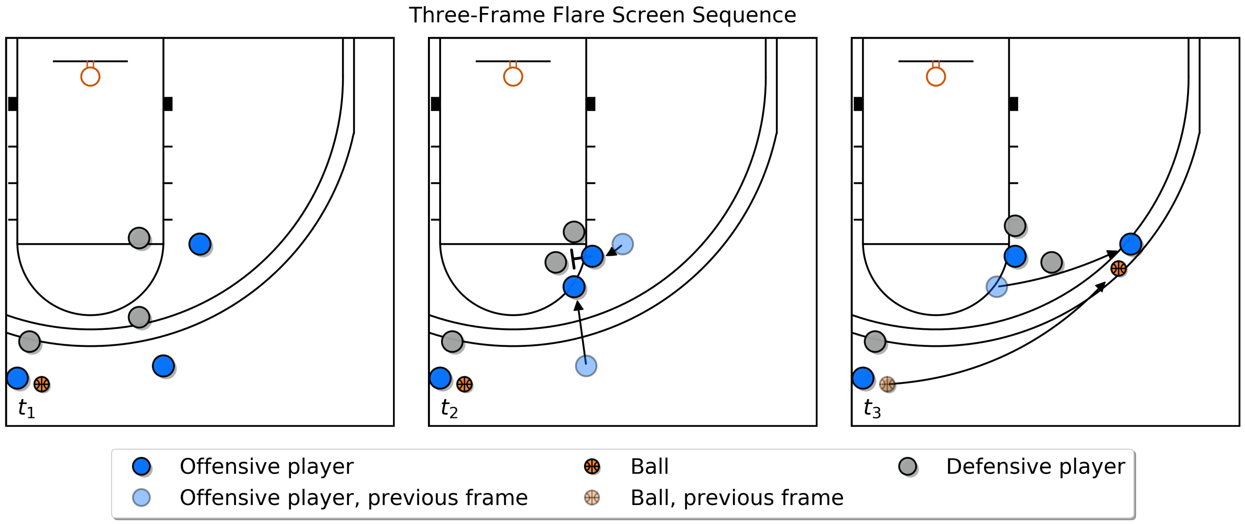

Figure 3.3: Three-frame sequence showing an example of a flare screen

Similar to a pin-down screen, a flare screen is an off-ball screen that happens with two offensive players with the intention of getting one of the offensive players open for a shot or an opening to drive to the basket. However, this screen is distinguished from a pin-down screen because of the orientation and position of the player setting the screen. For a flare screen, the orientation of the player setting the screen is pointed towards the sideline with the intention of sending the other offensive player towards one of the wings, as seen in the third frame of Figure 3.3. A common outcome of a successful flare screen is a jump-shot from one of the wings. In some scenarios, the flare screen is "curled" towards the basket, instead of going to the wing. As a side note, the BlueJays commonly have this "curling" flare screen action, which is reflected in Table 3.1.

3.1.4 Stagger Screen

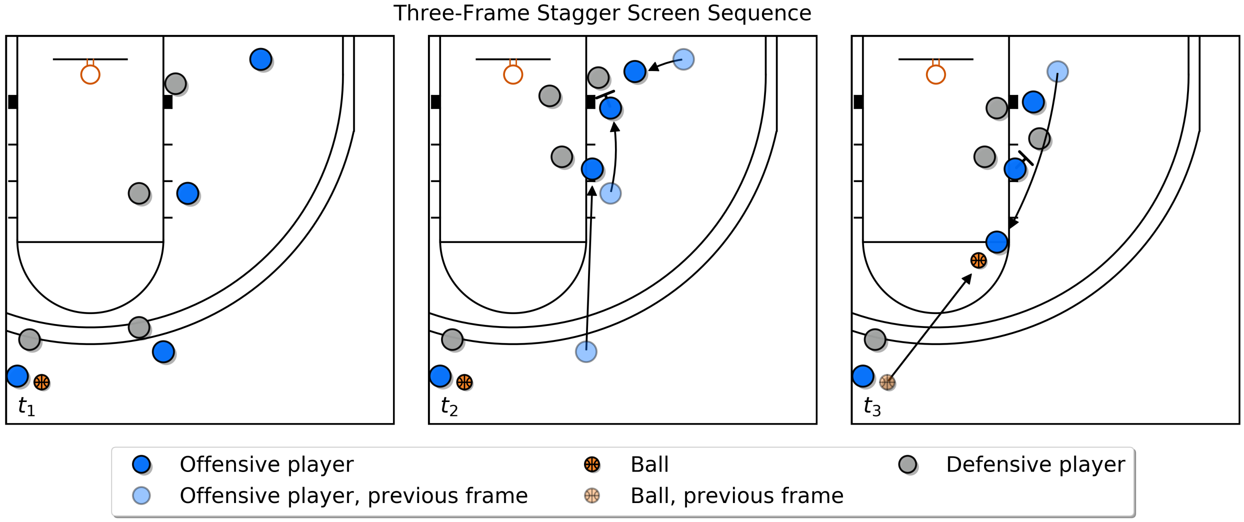

Figure 3.4: Three-frame sequence showing an example of a stagger screen

Similar to a pin-down screen, a stagger screen intends to get an offensive player open for a jump-shot or opening to attack the basket. However, the distinguishing feature for a stagger screen is that there are three total off-ball offensive players in the action. Instead of only one player setting the screen, there are two players setting successive screens. The benefit of a two-player screening sequence is to give the player coming off the screen an option to either use the second screen as intended or use the second screen as a flare screen. Similar to the pin-down screen, this creates an opportunity for the player setting the first screen to either get position to post up or create another action with the second player in the screening sequence, such as another pin-down screen. As explained in Section 3.2.1, there is low correlation between actions. Therefore even though the stagger screen consists of two separate pin-down screens or a pin-down followed by a flare screen, for the purpose of this paper, the classification of a stagger screen is independent of pin-down or flare screen actions.

3.1.5 UCLA Screen

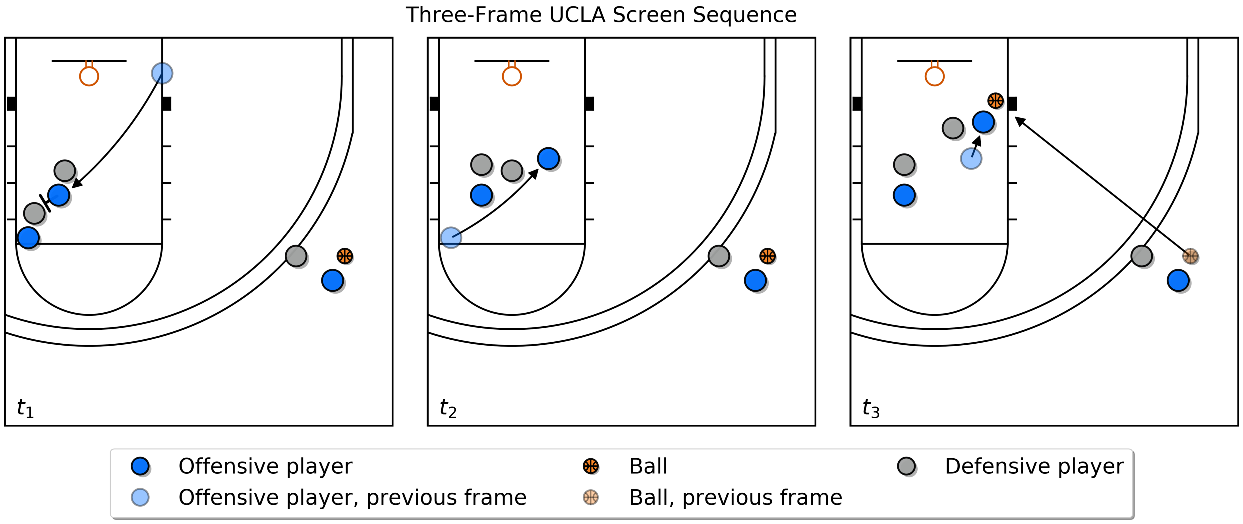

Figure 3.5: Three-frame sequence showing an example of a pin-down screen

Less common than pin-down, flare, and stagger screens, a UCLA screen is a screen with the intention of creating an open player near the basket. For the purpose of this paper, UCLA screens are also grouped in with up-screens or back-screens. This action involves two off-ball offensive players. The player who is receiving the screen is usually farther away from the basket and the player who sets the screen orients themselves away from the basket. Then, as seen in Figure 3.5, the player receiving the screen comes off the screen towards the basket. This allows two things to happen. First, the player who receives the screen gets position to post-up near the hoop. Second, it creates an opportunity for the offensive player who sets the screen to receive another action since their defender has to decide how they guard it.

3.1.6 Backdoor Cut

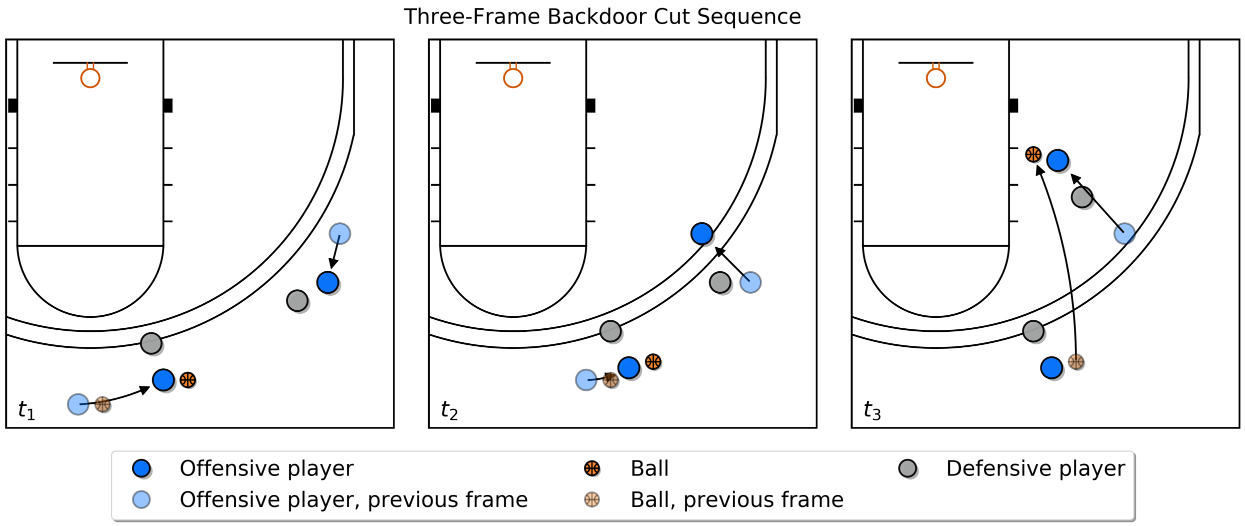

Figure 3.6: Three-frame sequence showing an example of a backdoor cut in which the ball-handler dribbles at the player cutting

The most common backdoor cut involves two offensive players, one of which is handling the ball. This push-action backdoor cut is where the ball-handler dribbles at a player on a wing, or in a corner, which triggers the other offensive player to cut towards the basket. If the defensive player who is guarding the off-ball player is overplaying, or attempting to deny a pass to that player, a backdoor cut can open a window for the offensive player being denied to receive the ball while moving towards a scoring area.

Additionally for our purposes, if the pass is not converted to the player cutting to the basket and retained by the ball handler, the action is still classified as a backdoor cut.

3.1.7 Rub-cut or Handoff

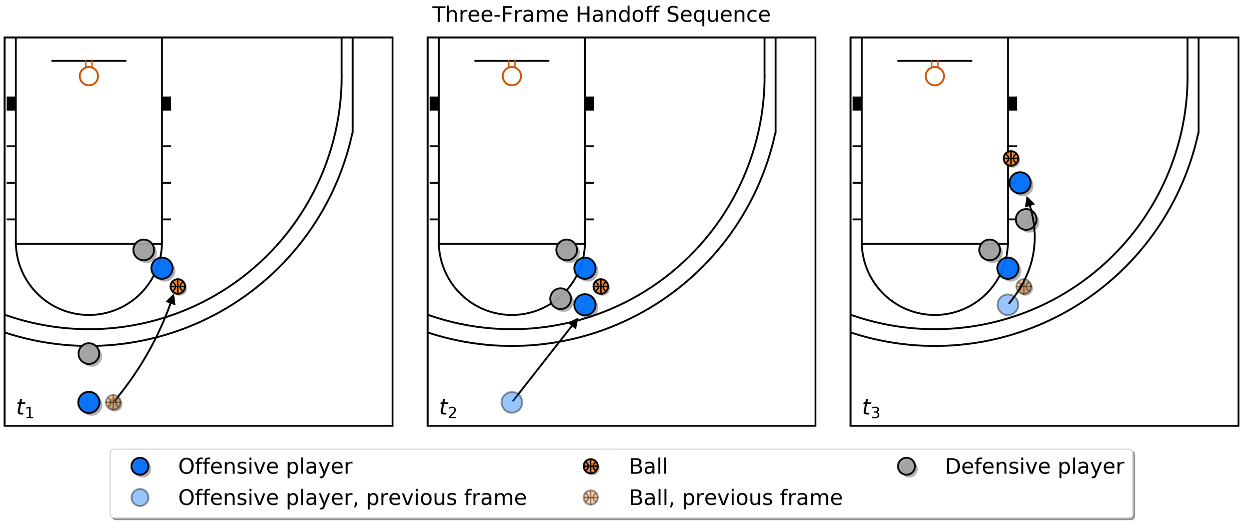

Figure 3.7: Three-frame sequence showing an example of a handoff or rub-cut in which the cutting player takes the ball

Similar to the backdoor cut, a rub-cut or handoff will be classified the same way regardless of which player ends up with the ball at the end of the action. Similar to a ball screen, this action happens with a ball-handler and an off-ball offensive player. However, in this scenario an off-ball player moves towards the ball-handler instead of the ball-handler moving towards a stationary off-ball player. Additionally, the ball-handler usually is oriented away from the basket and is inside the player who receives the handoff or performs the rub-cut.

3.2 Classification Models

To begin the discuss of classification methods, Table 3.2 marks the testing set accuracy of the optimal models for each action based on the appropriate amount of hyperparameter tuning.

	MODEL TEST SET ACCURACY		
ACTION	ANN	RNN	SVM
BALL SCREEN	0.907	0.978	0.785
PIN-DOWN SCREEN	0.808	0.925	0.722
FLARE SCREEN	0.831	0.957	0.736
STAGGER SCREEN	0.936	0.969	0.739
UCLA SCREEN	0.973	0.978	0.799
BACKDOOR CUT	0.873	0.953	0.711
RUBCUT/HANDOFF	0.972	0.986	0.698

Table 3.2: Testing set accuracy values from the best classification models for the seven common actions identified in a basketball possession

There are three main steps taken in order to create these classification models. First, the 929 possessions were hand-labeled with the actions described in Section 3.1. Next, the input data described in Section 2.2.6 is fed into classifiers built in TensorFlow [15] and scikit-learn [7]. Finally, these models are tuned based on choices of hyperparameters. In what follows, each section will describe the type of model, appropriate setup for the model, the hyperparameter tuning associated, and details about the results. For all the models, a softmax activation function is used to determine the output classifications and the ADAM [4] algorithm was use as the optimizer. In addition to hyperparameters, another common aspect of the Neural Network (NN) classifiers was the number of epochs and the learning rate. Generally, the learning rate was set at 0.005 and the number of epochs was between 50 and 100.

3.2.1 Feed-Forward Neural Networks

For classification with Artificial Neural Networks (ANNs), the approach involves building seven independent classifiers with one for each of the action. Previous work on multi-class, multi-label loss functions [6] mention the limitations of a pooled approach when there are a lot of unique class-label combinations and a small dataset. In our case, there are 173 unique combinations of actions which would need a larger dataset to combat these models overfitting. Addtionally, there is low correlation between action classifications, which helps justify using independent classifiers for each action.

As a refresher, the goal of Neural Networks (NNs) is to approximate a function from a set of continuous functions:

$$\mathcal{F} \in C(K) : K \in \mathbb{R}^d \rightarrow \mathbb{R} \tag{3.1}$$

This is accomplished by estimating the weights [CITE] using backpropagation. In this scenario, given a loss function $\mathcal{L}$, a set of weights $\vec{w}$, and a training set (X_i, y_i), $i = 1, ..., N$, the minimization is:

$$\min_{\vec{w}} \Big(\frac{1}{N} \sum_{i=i}^{N} \big[\mathcal{L}(h(X_i), y_i) \big] + \alpha ||\vec{w}||^2 \Big) \tag{3.2}$$

When training models for each action, data is balanced based on the number of unique actions that happen within each possession. This will help build networks that are more robust to being affected by any individual possession.

There are four primary hyperparameters tuned for the ANNs. First, the appropriate number of hidden layers was 3-5. Second, the classifier performed best when the architecture was a “diamond” shape. This means that one of the layers in the middle had the highest number of nodes compared to other layers. An example of this, based on number of nodes for a four-layer network, could be: 1200, 3600, 1800, 1600. The batch size, which determines how many instances are seen before updating weights, was optimal when there were 12-24 possessions in a batch. Finally, to prevent overfitting, associated dropout layers helped the ANNs perform the best when the dropout probability was between 5% and 10%.

For an example of hyperparameter tuning, Figure 3.8 shows an testing set accuracy for each action-classifier based on a 3-by-3 grid search on the first and second hidden layer sizes.

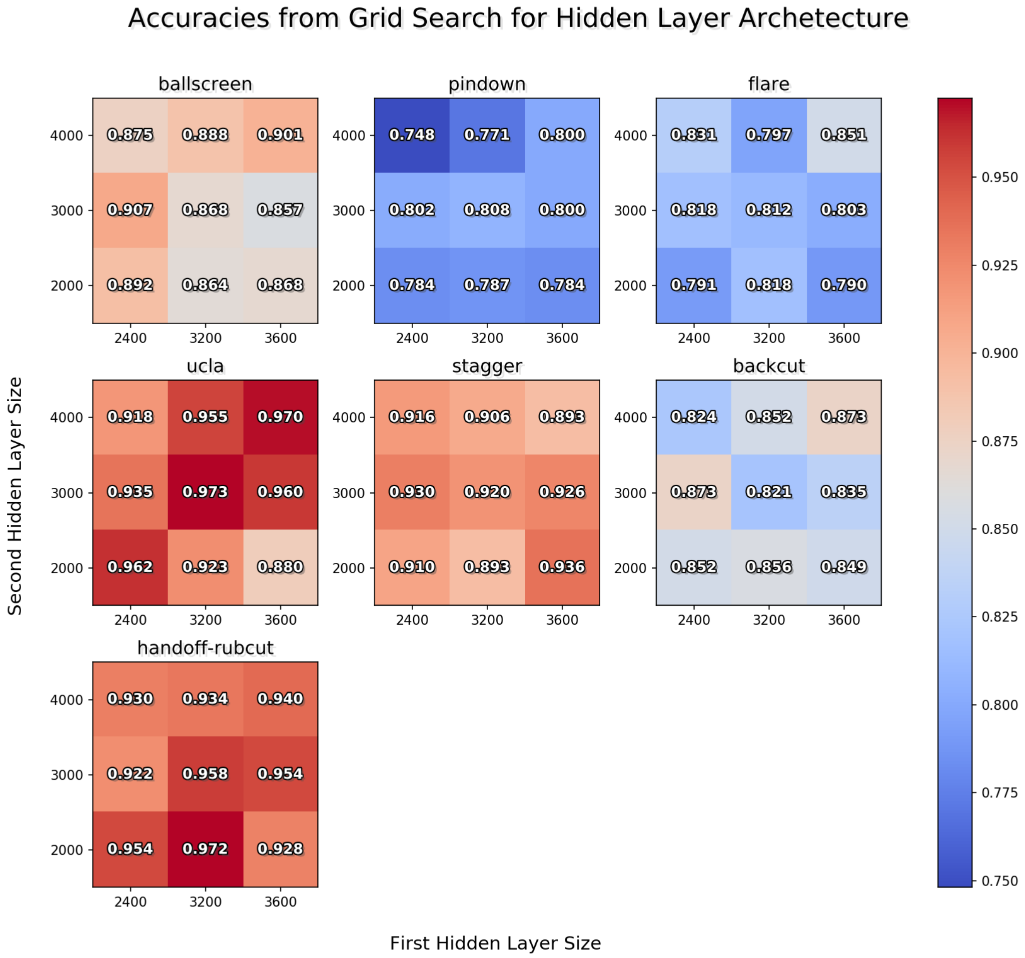

Figure 3.8: 3-by-3 grid search for feed-forward Artificial Neural Network (ANN) hidden layer architecture in each action-based classifier

3.2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are commonly used in scenarios where there is a time-based sequence of information. Since basketball possessions have a time-based dependence, RNNs are an applicable approach. With a similar loss function, RNNs aim to characterize these sequences using a set of Long Short Term Memory (LSTM) cells [3].

There are five hyperparameters that were investigated for RNNs. First, the best number of LSTM cells was around the max-length (in timesteps) of the possessions, which was 125. Next, the presence of a second LSTM layer slightly improved accuracy over models that just had one LSTM layer. Next, the optimal batch size was between 16 and 64 possessions. The models were most robust when there was 10% to 20% dropout and 10% recurrent dropout. Finally, there was no added accuracy from an additional dense layer after the layers of LSTM cells.

For an example of hyperparameter tuning, Figure 3.9 shows an testing set accuracy for each action-classifier based on a 3-by-3 grid search on the batch size and number of LSTM cells.

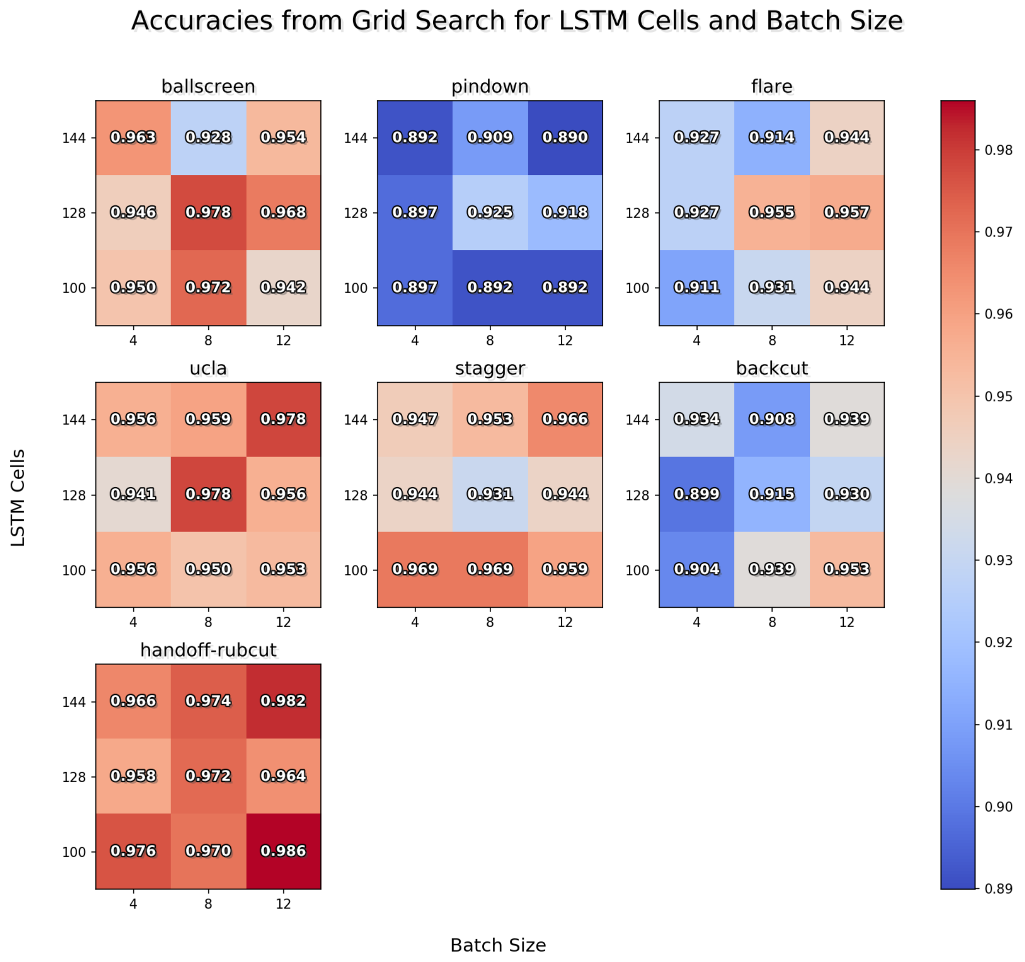

Figure 3.9: 3-by-3 grid search for optimal batch size and number of LSTM cells in each action-based RNN classifier

3.2.3 Support Vector Machine

Support Vector Machines (SVMs) are commonly used in scenarios to investigate the separability of the input data. For this case, the SVMs were set up with a radial basis function as the kernel function with a large value for the regularization term and a low slack penalty. As seen in Table 3.2, the SVMs fail to classify as accurately as NNs. This would help conclude that the data is difficult to separate and that the function we wish to estimate is not convex. Further, Figure 3.10 shows the cumulative percentage of variance explained by each of the first 100 Principal Components (PCs) from the possession data.

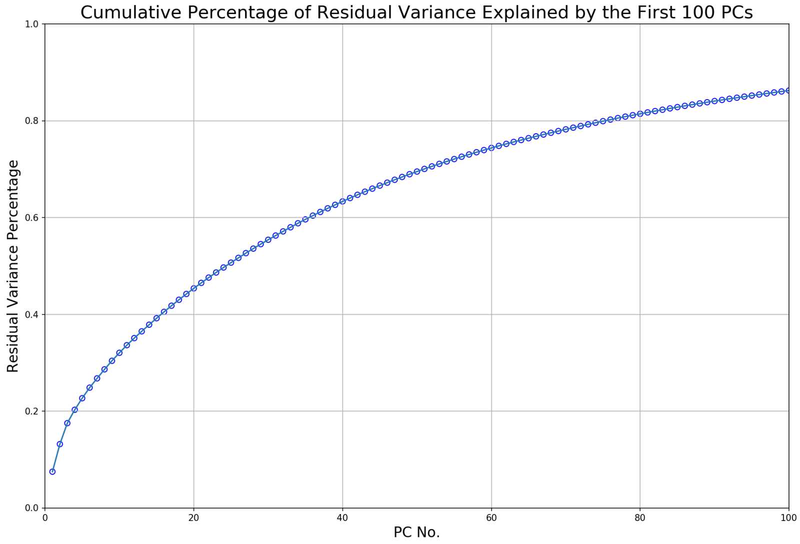

Figure 3.10: Cumulative percentage of residual variance explained by each of the first 100 Principal Components (PCs) from the possession-based input data

From analyzing the variance explained by the Principal Components (PCs), it makes sense that NNs outperform SVMs since the data does not seem to be easily separable.

3.2.4 Remarks

The high accuracy of the ball screen classifiers comes from the specific location that this action happens on the court. A reason why the classifiers run into trouble with the ball screen is that the defense, as mentioned in Section 3.1, can choose to react several different ways. To contrast, the pin-down screen classifiers struggle from the number of different places that a pin-down screen can happen on the court. However, the orientation of the screen is easy for the classifiers to pick up on.

Next, the flare screen has a high classification accuracy. This is from the location and orientation of the screen along with the frequency that the BlueJays use it. Variability in this action comes from whether or not the offensive player decides to curl the screen or take it to get open on the wing. In addition to the flare screen, the classifiers for the stagger screen do well because the stagger screen only happens in one unique place on the floor. Next, the UCLA screen has a distinct movement and orientation and therefore is classified at a high-rate. However, more data would be needed to generalize this since the UCLA screen is the second least common action. Next, the handoff or rub-cut is classified well since the action only happens in two places on the court. Variablity for the handoff or rub-cut could come from the choice that the ball handler has to pass off the ball to the other cutting offensive player. In addition to the rub-cut or handoff, variability in backdoor cut also comes from the choice of the ball-handler to deliver the ball to the cutting player. Additionally, the backdoor cut can be hard to classify because of the nature of some push-action offenses that passively push wing players through when being dribbled at.

The second notable result is that using RNNs, which inherently handle time-dependent data, outperform ANNs and SVMs. Specifically, the best RNNs beat the best ANNs in testing set accuracy for all of the actions. Another general note is that some error comes from plays that use, for example, two backdoor cuts and plays that use three backdoor cuts. More data would help determine the distinction in movement within these scenarios and help build models that can distinguish between classes when using softmax as the output activation. However, in general both NN methods are around 90% accurate and the SVMs are around 75% accurate in in classifying actions that happen within basketball possessions.

Chapter 4

Results of Applying Action Classification

There are two main applications of action classification within possessions. First, building classifiers with a high level of testing data accuracy can allow us to label previously unlabeled data. The second application is using these classifiers to examine the relationship between actions in a possession and their box-score outcome. Specifically in this section, we investigate possession efficiency for the BlueJays based on labeled action classification and predicted action classification in terms of Points Per Possession (PPP). PPP is the average number of points that a team scores during a possession. It is important to note that for the scope of this work, only offensive points-based outcomes were explored since it has the most direct application. After presenting efficiency, a discussion of the causality and potential confounding effects of a simplified metric will be discussed. This is followed with proposing future developments of the research in terms of possession efficiency metrics.

4.1 Labeling new data

The classification models in Section 3 show that with a high level of accuracy, seven common basketball actions can be classified. Although these seven actions are generally common for the entire game of basketball, using these particular models could be difficult for any one basketball team or set of basketball teams without further research. If there is reason that teams would play similarly at a certain level, such as the Amateur Athletic Union (AAU), NCAA, and NBA, models such as the ones built in this paper could generalize to classify possessions at these different levels.

Further investigation into this is beyond the scope of this paper but exists as an extension of this research in the future.

Ultimately, it is unreasonable to assume that each team will go through, label some or all of their data, and test the accuracy of the classification algorithms proposed. Therefore, that brings up the question: *how much data do you need label to get a decent long-run classifier?*

To answer this question, two approaches are taken. First, RNN models are trained on different sized of training sets and accuracy is analyzed. Then, a transfer learning approach [20] is taken to iteratively label more data and further classify actions. Both of these approaches are compared to the true action classification and respective PPP metrics.

Figure 4.1 compares, for a RNN with the same architecture, the testing set accuracy for different sizes of training data for each action classifier.

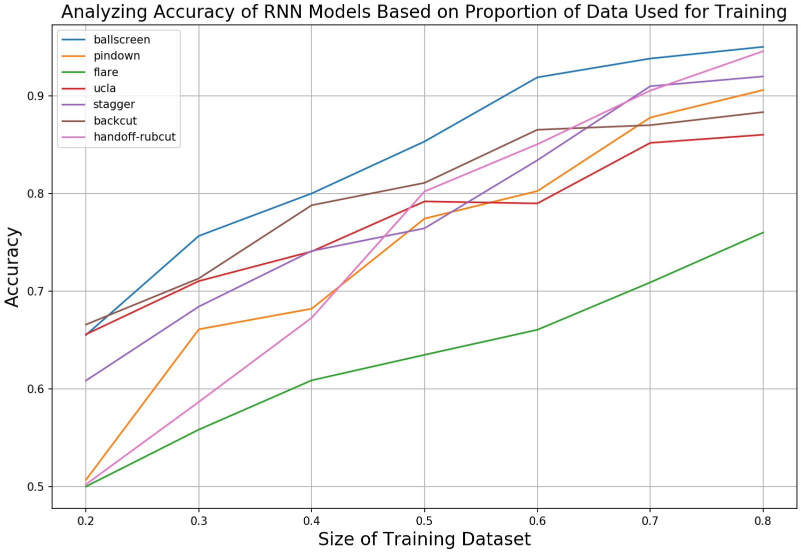

Figure 4.1: Accuracy comparing a RNN classifying with different training set sizes for each action

Intuitively from this, it is seen that with more training data, the classifiers are more accurate. However, it is also seen that with a reduced training set size, there is a high level of accuracy. This means that the classifiers could get a decent level of accuracy with only using half of the available training data. From this, we propose the following transfer learning algorithm.

This semi-supervised classifiers starts by taking 20% of the total labeled data and trains a RNN on this data for each action. After this initial training, a group of the size of 20% is taken, based on a K-Nearest Neighbors (KNN) algorithm and the initial classifier predicts the labels of this next 20% of the data. Then based on those predicted labels, the RNN weights and biases are retrained again. This would then be followed by predicting the labels for the next 20% based on KNN clustering of the input data. This is then repeated until the complete training data set is used up. This seems to work well as an application to basketball since the BlueJays continually work on the same plays or sets of plays. However, using just the BlueJays data to label another team might be more difficult. To demonstrate the accuracy of this KNN-based transfer learning algorithm, a testing set accuracy between this classifier and the best classifier from Section 3.2 are compared in Table 4.1

	MODEL TEST SET ACCURACY	
ACTION	RNN	TRANSFER
BALL SCREEN	0.978	0.845
PIN-DOWN SCREEN	0.925	0.801
FLARE SCREEN	0.957	0.844
STAGGER SCREEN	0.969	0.663
UCLA SCREEN	0.978	0.667
BACKDOOR CUT	0.953	0.785
RUB-CUT/HANDOFF	0.986	0.799

Table 4.1: Comparing the best RNN test accuracy with a transfer learning approach

From this, it is seen that the accuracy of a semi-supervised approach of classifying new data is varies compared to a full training set classifier. For the actions that happen more often for the BlueJays, the KNN-transfer learning algorithm does better than the actions that happen less frequently. There are three important things to note. First, this demonstrates as a proof-of-concept, that new possessions can be labeled using a semi-supervised approach with some level of accuracy. Second, future investigation is needed to determine at what number of total possessions (instead of proportion of possessions in the training set) is necessary to label before benefiting from the transfer learning approach. This will be investigated in a future line of research. Finally, the use of the labels from this classification technique is used for possession efficiency in Section 4.2 compared to the true possession efficiency.

4.2 Analyzing possession efficiency

With a focus of this paper being on creating analytics on a team-based level, we attempt to analyze the possession efficiencies for the BlueJays. The first part of analyzing possession efficiency will be to report in terms of Points Per Possession (PPP) based on the true action labels. This is broken down both for each individual action and the total number of actions that happen within a play. The second part of this section is to compare PPP between the true labels and the predicted labels for both our best classifiers from Section 3.2 and the transfer learning models from Section 4.1.

4.2.1 True individual action-based efficiency

First, Table 4.2 shows two efficiency statistics: Points Per Possession (PPP), Points Per Possession Without Action (PPPw/o). Points Per Possession is the average number of points the team scores on a possession with at least one of the corresponding actions. Points Per Possession Without Action (PPPw/o) is the average number of points the team scores on a possession without that common action. This will help show the contrast between the actions and see which actions produce higher scoring possessions.

	EFFICIENCY	
ACTION	PPP	PPPw/o
OVERALL	**0.82**	
BALL SCREEN	0.95	0.73
PIN-DOWN SCREEN	0.78	0.85
FLARE SCREEN	0.78	0.85
STAGGER SCREEN	0.77	0.83
UCLA SCREEN	0.75	0.83
BACKDOOR CUT	0.84	0.82
RUB-CUT/HANDOFF	0.96	0.78

Table 4.2: Points Per Possession (PPP) and Points Per Possession Without Action (PPPw/o) for the seven common actions identified in a basketball possession

From Table 4.2, it is seen that three actions, ball screen, backdoor cut, and handoff/rub-cut, produce higher PPP than when the action is not present. Expanding upon this, Table 4.3 shows PPP for combinations of these productive actions.

	EFFICIENCY	
ACTION	PPP	PPPw/o
BALL SCREEN & BACKCUT	0.93	0.82
BALL SCREEN & HANDOFF/RUB-CUT	0.94	0.81
BACKCUT & HANDOFF/RUB-CUT	0.89	0.82

Table 4.3: Points Per Possession (PPP) and Points Per Possession Without Action (PPPw/o) for combinations of productive actions.

This shows that the BlueJays are more productive on offense when they have a combination of a ball screen, backdoor cut, handoff or rub-cut rather than a UCLA screen, pindown screen, backdoor cut or flare screen.

Even though these metrics are useful, it is important to note that there is potential bias based on the usage of actions for the BlueJays. For example, the usage of a handoff and a ball screen could be more efficient since it could be a common multi-action play for the BlueJays.

4.2.2 True total action-based efficiency

At all levels of basketball, coaches tell players that movement and action help with scoring. Table 4.4 shows total action-based efficiency with two statistics: Points Per Possession (PPP) and Points Per Action (PPA). In this case for PPP, this table looks at the PPP for each number of actions seen in possessions. Points Per Action (PPA) is the average number of points that each action contributes to the score of the play. It is important to note that even though PPA is additive and does not have a good interpretation since there is a cap on the number of points scored in a possession, it is used to see the relative value having a certain number of actions within the possession.

	NUMBER OF ACTIONS						
ACTION	0	1	2	3	4	5	6+
PPP	0.85	0.86	0.88	0.87	0.93	1.17	1.23
PPA	0.390	0.355	0.289	0.232	0.203	0.209	0.268

Table 4.4: Points Per Possession (PPP) and Points Per Action (PPA) for each number of actions seen in a possession.

From this, it is seen that generally, the BlueJays PPP increases as the number of actions increases. There could be other factors at play here, such as using practice-based possessions or being in the presence of an offensive rebound [2], which both might increase the likelihood of scoring. Nonetheless, this shows that for the BlueJays, more actions within a play lead to more points. It is also important to note that confounding events, such as a turnover or offensive rebound, is necessary to investigate in order to address causality, which will be explored in future research. For example, does a turnover happen because a team decided not to run any action and isolate a player and that player turned the ball over or does a turnover come from the team not being able to effectively run a certain action? These questions could be answered by figuring out some movement-based context within the play, similar to suggestions in Stephen Shea's book, *Basketball Analytics: Spatial Tracking* [8]. However, it is generally a common understanding that more actions that happen within a basketball play lead to a higher likelihood of a better possession outcome.

4.2.3 Comparing true versus labeled efficiency

As an application to action-based classifiers for the BlueJays, the PPP metrics of the true labels and the predicted labels is shown in Table 4.5.

	EFFICIENCY (PPP)		
ACTION	TRUE	RNNs	TRANSFER
OVERALL	**0.82**	0.81	0.79
BALL SCREEN	0.95	0.94	0.99
PIN-DOWN SCREEN	0.78	0.73	0.65
FLARE SCREEN	0.78	0.75	0.70
STAGGER SCREEN	0.77	0.72	0.68
UCLA SCREEN	0.75	0.70	0.61
BACKDOOR CUT	0.84	0.84	0.77
RUB-CUT/HANDOFF	0.96	0.97	0.98

Table 4.5: Points Per Possession (PPP) comparing the true labels with predicted labels using classifiers for the seven common actions identified in a basketball possession

The takeaway from Table 4.5 is that for the BlueJays, comparing possession efficiency broken down between actions has two results. First, for the high-efficiency actions such as a ballscreen and rub-cut/handoff, from Section 4.2, show similar results for PPP as the true labels. Additionally, the overall PPP efficiency is similar between the true labels, predicted labels from RNNs, and predicted labels from the KNN-transfer learning approach.

Chapter 5

Conclusion

As analytics in basketball expands from the weight room to the front office to the court, there continues to be a need for answers in the right context. These advanced forms of analytics have the potential to help players develop and ballclubs win games. However, it is still important to harness the power of analytics in order to benefit all parties involved. The core of this power comes from the coach-player relationship. The ShotTracker system provides information for both players and coaches to benefit from. This research provides an expansion on the features produced in the built-in ShotTracker system, ultimately providing some additional insight into how our made-up team, the BlueJays performs in possessions based on the actions they decide to run. This information requires two steps.

5.1 Step 1: Classifying actions

In order to classify actions that happen within plays, the raw (x, y, z) tracking data is first pulled from the ShotTracker API. Then, it is processed in two steps. First, holes in the data are filled by a recursive midpoint rule that adds noise. Then, the data is projected onto a half-court axis, standardized, and reshaped into appropriate formats for the classification algorithms. Two classification methods are used for each action. First, feed-forward Vanilla Neural Networks (ANNs) produced testing set accuracy between **80% - 97%**. Second, Recurrent Neural Networks (RNNs), which rely on time-dependent data, produced testing set accuracy of **92% - 98%**. Finally, Support Vector Machines (SVMs) produced testing set accuracy of **69% -79%** and a look into

Principal Component Analysis (PCA) reveals, based on cumulative residual variance explained by the Principal Components (PCs), that the data is not easily separable. It is concluded that based on location-based tracking data, the seven common actions that happen within basketball plays can be classified.

5.2 Step 2: Possession efficiency

For the BlueJays, Points Per Possession (PPP) overall was 0.82 and increased to 0.95 with the presence of a ball screen, 0.84 with the presence of a backdoor cut, and 0.96 in the presence of a rub-cut or handoff. Additionally, to highlight the relationship between the number of total actions within the play and PPP, the BlueJays average 1.17 PPP when there were at least 5 actions performed. In contrast, with only one action, the BlueJays average 0.85 PPP. This information shows that there is a relationship between the actions that happen within a play and the points scored.

In addition to looking into the true value of PPP based on action labels, an estimated PPP is compared based on using the predicted labels from the best RNN models and a new KNN-based transfer learning algorithm. In terms of analyzing possession efficiency, the error between the RNN model labels and the true labels is **0.01 - 0.05** PPP. The error between the transfer learning labels and the true labels is **0.02 - 0.14** PPP. This shows that at some level, an appropriate semi-supervised approach can estimate PPP for a particular team.

5.3 The future of this research

In addition to the future improvements mentioned without the manuscript, there are three main future expansions of this research. First, in terms of data engineering, generative models could be explored to fill in large holes in a way that mimics natural movement. Second, other classification techniques such as Convolutional LSTM networks could be explored to classify these actions. In addition, instead of learning actions, there is room to learn plays, which are effectively sets of actions. This could require classifying actions, clustering plays, and using an ensemble method to identify what play in particular it is. Finally, efficiency calculations could be expanded to include time-dependence (i.e. first versus fourth quarter) or control for other variables, such as lineups or other statistics like offensive rebounds.

Bibliography

[1] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In ESANN, 2013.

[2] Kirk Goldsberry. Sprawlball: A Visual Tour of the New Era of the NBA. Houghton Mifflin Harcourt, 2019.

[3] Keras. Lstm layer. `https://keras.io/api/layers/recurrent_layers/lstm/`, 2020.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[5] Michael Lewis. Moneyball: The Art of Winning an Unfair Game. W. W. Norton & Company, 2003.

[6] Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. Large-scale multi-label text classification - revisiting neural networks, 2013.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[8] Stephen Shea. Basketball Analytics: Spatial Tracking. 2014.

[9] ShotTracker. Track 70+ stats instantly with shottracker. `https://shottracker.com/videos/272686985`, 2018.

[10] ShotTracker. Shottracker. `https://shottracker.com/`, 2019.

[11] ShotTracker. About us. `https://shottracker.com/about-us`, 2020.

[12] ShotTracker. Shottracker developer. `http://developer.shottracker.com/`, 2020.

[13] Second Spectrum. Our work. `https://www.secondspectrum.com/ourwork/leagues-media.html`, 2020.

[14] TensorFlow. Masking and padding. `https://www.tensorflow.org/guide/keras/masking_and_padding`, 2020.

[15] TensorFlow. Sequential api. `https://www.tensorflow.org/guide/keras/sequential_model`, 2020.

[16] Changjia Tian, Varuna De Silva, Michael Caine, and Steve Swanson. Use of machine learning to automate the identification of basketball strategies using whole team player tracking data. 2020.

[17] Kuan-Chieh Wang and Richard S. Zemel. Classifying nba offensive plays using neural networks. 2016.

[18] L. Jon Wertheim and Toby Moskowitz. Scorecasting: The Hidden Influences Behind How Sports Are Played and Games Are Won. Crown Publishing Group, 2011.

[19] Daniel Wilco. College and nba basketball's biggest rule differences. `https://www.ncaa.com/news/basketball-men/article/2019-09-26/college-and-nba-basketballs-biggest-rule-differences`, 2019.

[20] Xiaojin Zhu, Andrew B. Goldberg, Ronald Brachman, and Thomas Dietterich. Introduction to Semi-Supervised Learning. Morgan and Claypool Publishers, 2009.