Utilizing RNN Architectures to Evaluate the Success of an NBA 3-Point Shot

Archit R. Kambhamettu
Department of Computer Science
University of Maryland
College Park, MD 20740
architk@umd.edu

Saurabh V. Gadre
Department of Computer Science
University of Maryland
College Park, MD 20740
sgadre12@terpmail.umd.edu

Eamon Weingold
Department of Computer Science
University of Maryland
College Park, MD 20740
eamonrw@terpmail.umd.edu

Brendan LaPuma
Department of Computer Science
University of Maryland
College Park, MD 20740
brendanlapuma@gmail.com

Rohit Mukund
Department of Computer Science
University of Maryland
College Park, MD 20740
rmukund1@terpmail.umd.edu

Abstract

Our ultimate goal for this Deep Learning Final Project was to utilize both the SportVU dataset [1] which contains positional data for NBA players on a 2D plane for the 2015-2016 season and a Basketball-Reference statistical dataset [4], the corresponding players' NBA game statistics, to determine the probability that a given three-point shot was made. To do this, we leveraged the aforementioned datasets as well as other sources containing play-by-play data [3] to create our own dataset, with each datapoint consisting of 10 frames (1 second = 1 frame) before a shot was attempted. Each frame contained the corresponding relative player distances, offensive statistics (for each player on offense on the court for that play), and defensive statistics (for each player on defense on the court for that play). We furthermore evaluate the efficacy of three deep learning architectures: Recurrent Neural Networks, Gated Recurrent Units, and Long Short Term Memory models on our curated dataset, and surpass all current benchmarks for this task. Following the completion of our model, we explore different real-world avenues where our model could be applied. First, we see that we can display the names of key players on the court by mapping the SportVU dataset to broadcast footage, and can display whether the shot will go in or not on the given frame. This will aid novice viewers in learning what possessions are good, and who are good three-point shooters. Another avenue we explore is by generalizing this model to work on all basketball game footage, done by translating the player positions from the broadcast footage to the SportVU coordinate system and then feeding it into our model, presenting a pipeline to encourage future researchers to tackle this task. Thus, we developed a system to successfully classify whether a given NBA three-pointer would result in a made or missed basket, and discuss real-world applications and generalizations.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

1 Motivation

The motivation behind the project is to try to introduce some form of empiricism to further our understanding of decision making in basketball. By utilizing deep learning, we properly quantify the optimality of a given three point possession. This line of research could be useful for various reasons. First, if implemented in an NBA game, it could provide useful information for a novice viewer to understand what a "good" shot or "bad" shot is. For instance, certain players taking slightly contested shots might be interpreted as a higher quality shot than another player taking an open shot. Thus, by seeing the percentage that a tree-point shot results in a make, a novice viewer would have a stronger understanding of potential player performance and the quality of a given possession. Similarly, coaches can determine whether their teams are attempting high quality three pointers throughout the course of the NBA game. While the optics might suggest that the team is receiving high quality opportunities, implementing a data driven approach that incorporates the nuances of the game could provide additional insight and lead to adjustments. Thus, teams would have the opportunity to devise a data-driving gameplan. To accomplish this motivation, we divide our overarching task into three main components: dataset creation, model architecture development, and visualization/generalization of results. Our dataset incorporates specified player distances for each of the 10 players on the court (further explained in the Datasets and Prepossessing Section), as well as each of their corresponding offensive/defensive NBA game statistics. As the primary scope of the project was to leverage deep learning to gain data-driven insights into NBA games, for the probability generation (and thus binary classification) of the made or missed shot, we focused only on leveraging deep learning architectures, rather than other models such as Random Forest, Logistic Regression, KNN, and other non-neural architectures. As a given basketball possession can be interpreted as a continuous time series sequence in seconds, where the previous "frame" affects the result of the next, we opted to utilize the RNN, LSTM, and GRU architectures due to their incorporation of such time series data in the model and determine their performance on the task at hand. As shown by our results, these models perform much higher than the current baselines discussed in the Baselines and Relevant Papers section. For the visualization component of this task, we initially assumed that the direct translation from the SportVU Dataset to the broadcast footage would be consistent for all games and frames; however, due to the varying feeds of broadcast footage (ie. broadcasts change throughout the game from zoom in/zoom out/ shifts and change across games from different locations/broadcasting networks), we decided to train a Yolo-V8 model on a custom dataset to do keypoint localization on the NBA court and then perform homography to properly translate the SportVU Dataset to broadcast footage. We also train another Yolo-V8 model on a custom dataset to perform player detection on the video footage to set the stage for future generalizations of our current model with broadcast footage data. Our current models surpass all current benchmarks for shot quality detection, as relayed in the sections below.

2 Baselines and Relevant Papers

The SportVU data set has been used to solve numerous tasks in the "basketball + deep learning" community. In [10], author Goldsberry utilized the dataset to assess NBA players' shooting abilities, presenting an ensemble of analytic techniques to describe the shooting tendencies of players with spatial data, thus paving the way for the incorporation of player localization in advanced basketball analytics. Following the official partnership between the NBA and SportVU to release their datasets to the public, a series of publications followed in this line of research. In Dan Cervone et. al's work [8], the authors coin the term "Expected Possession Value (EPV)," representing a score on how likely a given decision will result in a positive outcome for the team. The paper focuses on statistical models to maximize EPV in their research [8] similarly evaluated decisions in a given basketball game via a stochastic process model, calculated the Expected Points produced for a given possession, as well an evaluation of every action (pass, shoot, dribble, etc) within a given possession. While these approaches were statistically oriented, rather than Deep Learning oriented, there seems to be a promising opportunity to explore this schema within the context of a Deep Neural Network. Another work [13], leveraged the RNN architecture to determine whether shots from 2-8 ft from the basket would be made or missed. The paper focused on ball trajectories pulled from the StatVu dataset, and extrapolated well onto the 2014-2015 season. A term paper in machine learning [15] leveraged machine learning models to classify whether a shot would be missed or made, with XGBoost performing the best with a 68 percent classification accuracy. While this data was pulled

from Kaggle, we find it to be the best baseline benchmark accuracy for the task of three point shot make/miss classification. In [14], author Sliz described the important features that would aid in the three-point shot making classification task, emphasizing the importance of player distances to the task. While not necessarily the core of our project, much work has been done on NBA player detection. [7] leveraged the Yolov2 model for this task achieving reasonable results, while [9] utilized color segmentation and a HOG detector to detect both the players and key points on the court. Additionally, [11] created an end-to-end solution for classifying key points on the court, detecting players, tracking players, and classifying players with the purpose of counting statistics from broadcast footage. We build on these works to develop our model architecture, as detailed below.

3 Datasets and Preprocessing

The data collection process for our model was extensive and spanned multiple datasets, but it resulted in a large, clean, and easily testable dataset that maintained its stability over multiple iterations of the neural architecture. The primary focus of our Recurrent Neural architecture is the moment-by-moment player positioning on the NBA court, so most of the data collection process was centered around SportVU Basketball [1]. The SportVU dataset is collected from a real-time camera-based tracking system installed in every basketball arena in the NBA league. The dataset holds important metadata about every single player on the court such as their name and unique player ID, along with their x-y position on the court. We collected SportVU data for the 2015-2016 NBA season from a pre-existing GitHub repository [2] and repurposed it for our model.

This information is repeated every time an updated position is tracked, which occurs approximately every two to three-hundredths of a second. This has the unfortunate consequence of extremely redundant data, and, despite only including data from the 2015-2016 season in the training set, the entire SportVU dataset contained over 60 gigabytes of data. The first step in the data preprocessing pipeline was to remove the redundancy and extract only the relevant information for our model. Firstly, to prevent oversaturating the neural model with a plethora of minuscule time-series steps containing only slight shifts in player movement, we opted to exclude any data whose game clock was not as close as possible to the nearest whole second. This singular change significantly reduced the file size and allowed for significant speed-ups for all subsequent data processing steps.

Although the SportVU dataset, at this point, gave us access to player movement data for every second of an NBA game, we were only interested in the movement of players leading up to a 3-point attempt. Thus, we merged our SportVU dataset with the shotdetail dataset from the official NBA stats website, collected by a third party on Kaggle [3], which contains information for every single NBA play. These stats include the shot type, distance, and whether or not the shot was a success. We filtered to only include 3-point shot attempts. Additionally, each play has a timestamp for when it occurred during the game, and we duplicated each play 9 times to include the 9 previous seconds leading up to the shot, for a total of a 10-second window. In this manner, performing an inner merge on the SportVU dataset and the shotdetail dataset provided us with player movement data for 10 seconds leading up to a shot in addition to a "label" representing whether or not the shot went in.

At this point, there was enough numerical information to feed into a neural model, but we needed to deal with the fact that x and y locations on a basketball court would not interface well with a neural model, as higher or lower coordinate values do not necessarily correspond to higher or lower signals. Our solution was to convert these coordinates into distance metrics, centered around the 3-point shooter. For the shooter, their distance to the basket is recorded, and for the remaining 9 players on the court, we calculated their distance to the shooter and the basket. The order in which these players appear in our dataset is decided by which team they belong to (with the offense appearing first), followed by their distance to the shooter (closest appearing first). Maintaining this exact order was essential for data consistency when adding player statistics.

While player movements alone can be quite revealing when considering the success of a play, they do not consider what kind of players are on the court at the time. We decided to allow our model to learn much more nuanced information by introducing player-specific statistics for every player on the court. We collected offensive [4] and defensive [5] player statistics from basketball-reference and merged them with our player movement dataset with the help of an ID conversion database [6] that ensured parity between players on official NBA datasets and basketball-reference datasets. These additional datasets allowed us to incorporate the following offensive statistics for the 3-point shooter:

total games played during the 2015-2016 season, average minutes per game, average shot distance, 3-point attempt percentage, and 3-point success percentage. For all 5 defensive players on the court, we incorporated the following defensive statistics: total games played during the 2015-2016 season, average minutes per game, steals per game, blocks per game, and defensive box score plus-minus per 100 possessions.

After combining all of this information into one dataset and removing any gaps in the data that did not form a complete 10-second time series, we were left with a complete set of 23,669 unique NBA events each containing a 10-second time series of distance data and offensive/defensive stats for a total of 49 features every second. We settled on a training-validation-testing split of 75 percent, 20 percent, and 5 percent when training the recurrent network.

4 Model Architecture and Hyperparameter Tuning

After collecting and cleaning our data, our initial goal was to make the data usable by a machine-learning model. We normalized all the data categories and removed all the features that would not be relevant to the model's classification (player/team/game IDs, game clock, etc). We also chose to remove SportVU instances that occurred between seconds to have a more uniform distribution between instances that we fed into our model. After some experimentation, we used ten seconds (ten instances) leading up to a shot for our neural classifier. An example of a normalized feature vector of the first time instance in a list of 10 is shown in Figure 1.

Since our data was split into time series instances of length 10, we initially used a Recurrent Neural Network (RNN) as our primary classifier. This would allow us to feed all ten instances one at a time through the model, all correlating to the same shot attempt. As an initial starting point for our architecture, we followed a PyTorch RNN tutorial [12] intended for language classification and modified its functionality to work with our 3-point shot classification task. Firstly, we reduced the model's complexity to 1089 parameters to reduce the risk of overfitting our training dataset and improve training time. Then, we changed the dimensionality of the output layer to one so the model could be used for our binary classification task. Finally, we switched the activation function of the output layer to a sigmoid function so the one-dimensional output of the RNN can be interpreted as a probability. Furthermore, we implemented a data loader that batched our training data to further improve training time and optimize the process for GPU computations. After optimizing the hyperparameters for our RNN, we also attempted to use a Long Short-Term Memory (LSTM) model with 1969 parameters, and a Gated Recurrent Units (GRU) model with 3233 parameters. These architectures are alternative versions of RNNs that can also be used for time-series classification tasks. However, after extensive experimentation with these alternate architectures, we were unable to notably improve upon the results of our initial RNN. We were also extremely limited by computational power, since we could not use or reserve any GPU time on Google Colab, and were forced to perform all of our training and testing on the free and fairly slow T4 GPUs. While we did find a workaround that allowed us to use local hardware on Colab via Docker containers, using a laptop GPU was only a mild improvement over the free hardware offered by Google.

```
[ 1.2435e-01,  2.7686e-01,  2.4873e-01, -5.8280e-01, -8.9453e-01,
 -3.8404e-01, -8.9344e-02,  8.0000e-01, -2.0949e-01, -5.1602e-01,
 -2.6508e-01,  6.8067e-01,  1.4699e-01,  1.3296e-02,  8.0472e-01,
  3.3391e-01, -2.3504e-01, -3.6794e-01, -1.7649e-01,  4.7788e-02,
 -5.7393e-01,  8.7611e-01, -5.3125e-01, -2.3233e-01,  3.2406e-01,
  8.0666e-01, -2.9027e-01, -6.0113e-01, -3.7027e-01,  3.8767e-01,
  5.0398e-01, -8.2888e-01,  5.0993e+00,  2.1165e+00,  7.0817e-01,
  6.6528e-01,  2.0069e-02, -5.8183e-01,  5.9244e-01,  4.4729e-01,
  6.9012e-01,  3.2212e-01, -2.2047e-01, -3.4000e-01, -9.6254e-01,
 -6.7545e-02, -1.0825e+00,  1.1830e-01,  5.3404e-01],
```

Figure 1: Normalized Feature Vector Example

5 Results

Pictured below are the training results from all three models that we used. While we were cautiously optimistic that a model based on time-series data could classify 3-point shot attempts based on the statistics and relative distances between players, we were pleasantly surprised by the results. The RNN peaked at around 80 percent accuracy, the LSTM managed to reach 76 percent accuracy, and the GRU got 78 percent accuracy. These accuracy results were the maximum values achieved after multiple iterations through the entire training process, as the values tended to vary around 3-4 percent each time.

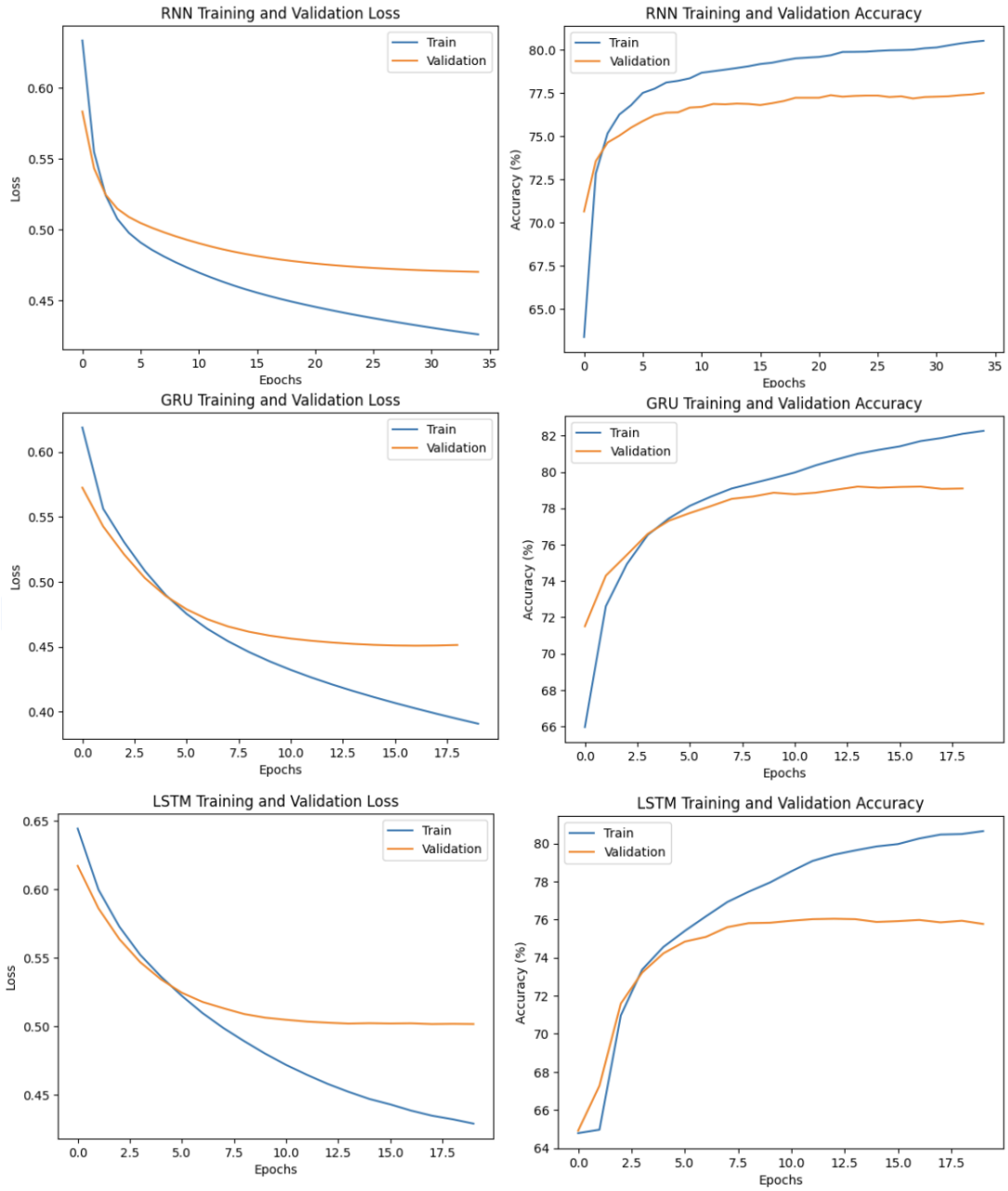

Figure 2: Loss and Accuracy Plots for Training and Validation Phases for the Three Models Above

6 Visualization

In order for this line of research to be valuable to the common audience, we decided to display whether the shot will go in or not on the corresponding broadcast footage of an NBA game, with the corresponding names on the court. This would potentially aid novice audiences in understanding the complexity of the game, which players are good three-point shooters, and add additional insight to their viewing experience. To do this, we first create a reliable mapping between the SportVu dataset coordinate system and the corresponding broadcast player system.

We performed this coordinate mapping by first detecting the keypoints on the court. Because this is not a common task in the object detection community, we decided to train a custom model to solve this task, using the dataset, consisting of 942 images and 11 classes, each detailing specific keypoints on the court such as "Left Bottom Left Paint", detailed in [16]. As this was not the primary research question for this project, we opted for the state-of-the-art, out-of-the-box Yolo-v8 object detection model. Primarily due to its ability to its ability to detect small objects, and our discussion of the model in class, we felt it useful for this task. Thus we trained our model on the aforementioned dataset for 175 epochs, with batch size of 16, working with images of size 640x640x3. Figure 3 displays the PR curve for the model, indicating that it performs very well on the given dataset. When taking "in the wild" test samples (ie YouTube highlight clips not found in the dataset), we found that while the detection of the keypoints exceeds expectations, the classification of the keypoints as a combination of right/left and top/bottom can be improved. This is demonstrated in Figure 4, which is a sample test image run through the model.

By successfully detecting these keypoints, we can freely translate between the SportVu dataset and any broadcast footage of a basketball game. We do this via homography, a computer vision technique to translate coordinate spaces. After graphing the x and y positions of the NBA players from the SportVu dataset across numerous games, overlaying the points on top of an NBA court image, we noticed that the coordinate system is consistent across all these games. Thus, we extracted the coordinate points for the 4 corners of the paint for each left/right viewpoint, as our points for homography calculation. We see that the model can thus overlay the player names underneath the player via the geometric calculation to convert the SportVU coordinates to the broadcast footage video, as seen in Figure 5.

Thus, we can now successfully determine whether a shot goes in or not and label the players in a given broadcast for a given frame, thus providing ample insight into the game for viewers of all level, for all games in the SportVU Dataset.

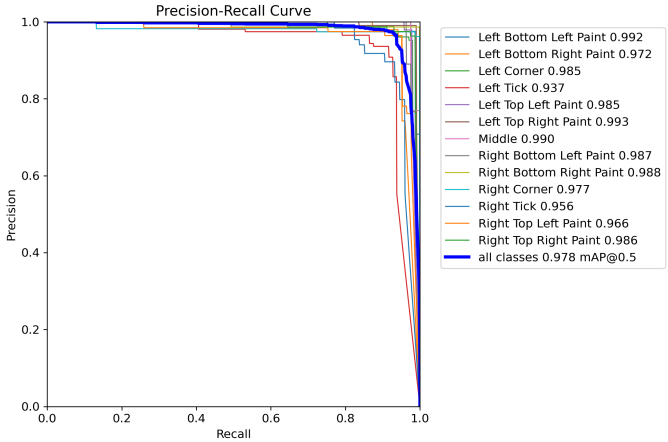

Figure 3: PR Curve of Keypoint Detection Model

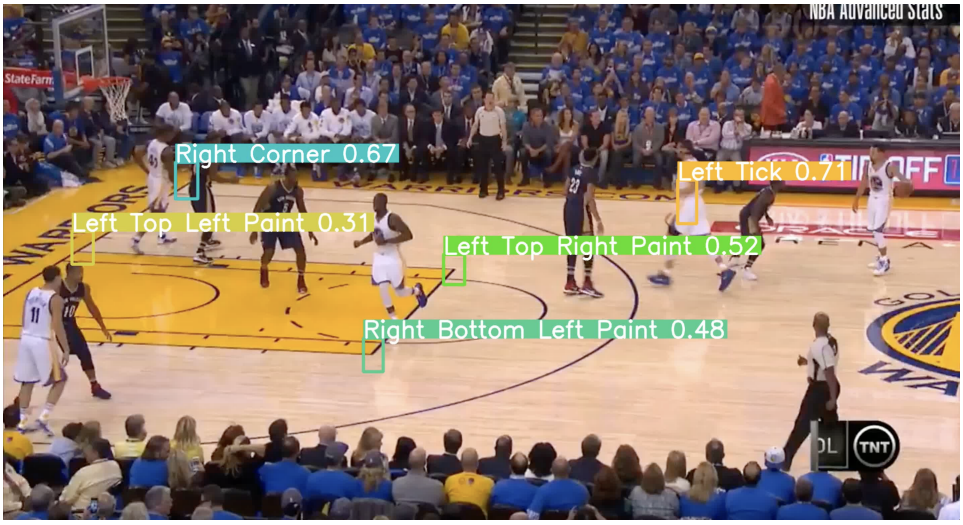

Figure 4: Sample Test Image of Keypoint Detection Model

Figure 5: Full Overlay of Player ID + Probability of Shot

7 Model Generalization and Future Work

Now that we have successfully determined whether a shot would be made or missed for a given SportVU dataset, we introduce a potential pipeline for future researchers and provide a baseline start to generalize our model to predict whether any given basketball game shot will be missed or made.

To make this model more generalizable to future applications, it is necessary for it to work on all games and coordinate systems, not just the SportVU dataset. Because our model is trained on the SportVU dataset, it expects relative distances to be calculated in the SportVU coordinate system, which is currently unknown to the common user. As broadcast footage of nearly all NBA (and other leagues') games is readily available, we felt that determining the coordinate mapping from one space to the other was the most practical approach to generalizing this model to all levels of play. To do this, we first detect NBA players on the court, then leverage the same homography pipelined mentioned in the previous section.

The first step for this pipeline is to detect the players on the court, which we did via a custom-trained YOLO-v8 object detection model. We utilized the YOLO architecture due to the aforementioned reasons and our in-depth discussion of the model in class. We trained on the dataset found in [17] (311 images of NBA games, with 3 classes of players, hoop, and ref) for 175 epochs with a batch size of 16 and the same image dimensions (64x64x3) as above to allow the same images to be passed into both models. Figure 6 displays the results (PR Curve) below. We see that the model has a 0.92 mAP@0.5 indicating a good model; however because our dataset was quite small for this task, we evaluated our player model by running it on a series of test videos extracted from YouTube clips. For

those videos, we see that while the model performs fairly well, there is still room for improvement, thus leaving room for future development in this area. We display a sample image of our model in Figure 7.

With the players on the court now detected from the broadcast footage, their positions can now be translated to SportVU coordinates by using the homography pipeline detailed in the Visualization section. Thus, these new rectified positions can be passed into our model to achieve the proper classification.

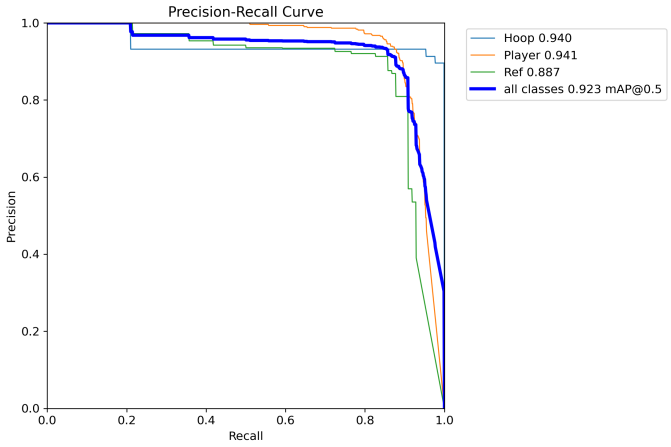

Figure 6: PR Curve of Keypoint Detection Model

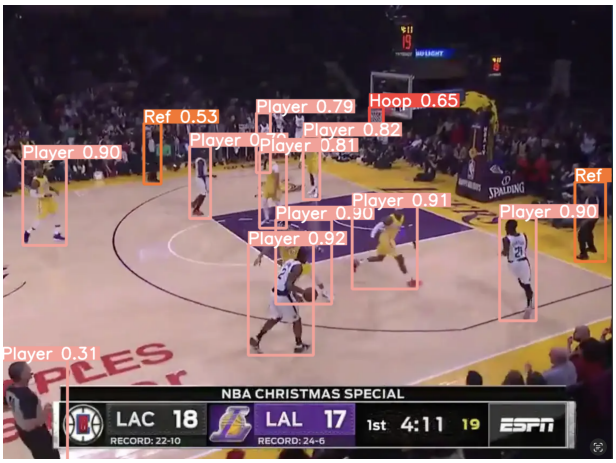

Figure 7: Sample Player Detection Image

However, it is important to note that while determining this mapping is the first step in generalizing the model, classifying who each player is to provide the complimentary stats would be necessary for the success of the model. Although we did not attempt to tackle this problem for this project, we see this as an interesting line of research for the future, as first, the proper team would need to be classified (offense vs defense) then the player would need to be segmented from that. Thus, this yields a promising direction for the future of this field.

References

[1] SportVU Basketball. (n.d.). Stats Perform. Retrieved May 9, 2024, from https://www.statsperform.com/team-performance/basketball/optical-tracking/

[2] Linou, K. (2016, September 19). NBA Player Movements. GitHub. Retrieved May 9, 2024, from https://github.com/linouk23/NBA-Player-Movements

[3] Shufinskiy, V. (2023, March). NBA play-by-play and shotdetails data (1996-2022). Kaggle. Retrieved May 9, 2024, from https://www.kaggle.com/datasets/brains14482/nba-playbyplay-and-shotdetails-data-19962021

[4] 2015-16 NBA Player Stats: Shooting. (n.d.). Basketball Reference. Retrieved May 9, 2024, from https://www.basketballreference.com/leagues/NBA$_2$016$_s$$hooting.html$

[5] 2015-16 NBA Player Stats: Advanced. (n.d.). Basketball Reference. Retrieved May 9, 2024, from https://www.basketballreference.com/leagues/NBA$_2$016$_a$$dvanced.html$

[6] Blechner, D. (2020, January 30). NBA Player and Team ID Database. GitHub. Retrieved May 9, 2024, from https://github.com/djblechn-su/nba-player-team-ids

[7] Acuna, D. (n.d.). Towards real-time detection and tracking of basketball ... University of Toronto. https://www.cs.toronto.edu/ davidj/projects/towards$_r$$eal_t$$ime_d$$etection_t$$racking.pdf$

[8] Cervone, D., D'Amour, A., Bornn, L., Goldsberry, K. (n.d.). A multiresolution stochastic process model for predicting ... arxiv. https://arxiv.org/pdf/1408.0777.pdf

[9] Cheshire, E., Halasz, C., Perin, J. K. (n.d.). Player tracking and analysis of basketball plays. Stanford. https://web.stanford.edu/class/ee368/Project$_S$$pring_1$$415/Reports/Cheshire_H$$alasz_P$$erin.pdf$

[10] Goldsberry, K. (n.d.). Courtvision: New Visual and spatial analytics for the NBA. MIT Sloan Sports Analytics Conference. https://www.sloansportsconference.com/research-papers/courtvision-new-visual-and-spatial-analytics-for-the-nba

[11] Lu, W.-L., Ting, J.-A., Little, J. J., Murphy, K. P. (n.d.). Learning to track and identify players from broadcast sports ... University of British Columbia. https://www.cs.ubc.ca/ murphyk/Papers/weilwun-pami12.pdf

[12] Robertson, S. (n.d.). NLP from scratch: Classifying names with a character-level RNN¶. Pytorch. https://pytorch.org/tutorials/intermediate/char$_r$$nn_c$$lassification_t$$utorial.html$

[13] Shah, R. C., Romijnders, R. (n.d.). Applying deep learning to basketball trajectories. arxiv. https://arxiv.org/pdf/1608.03793.pdf

[14] B. Sliz. An investigation of three-point shooting though an analysis of nba player tracking data. 2016.https://arxiv.org/pdf/1703.07030

[15] B.Meehan. Predicting NBA Shots (n.d) https://cs229.stanford.edu/proj2017/final-reports/5132133.pdf

[16] nba-court$_d$$ataset, NBACourtDataset(OpenSourceDataset).betatracker, https://universe.roboflow.com/betatracker/nba-court, fromRoboflowUniverse.$2023

[17] nbavision$_d$$ataset, NBAVisionDataset(OpenSourceDataset), NBAVision, https://universe.roboflow.com/nbavision/nbavision, fromRoboflowUniverse,$ 2024