PlayCaller.ai

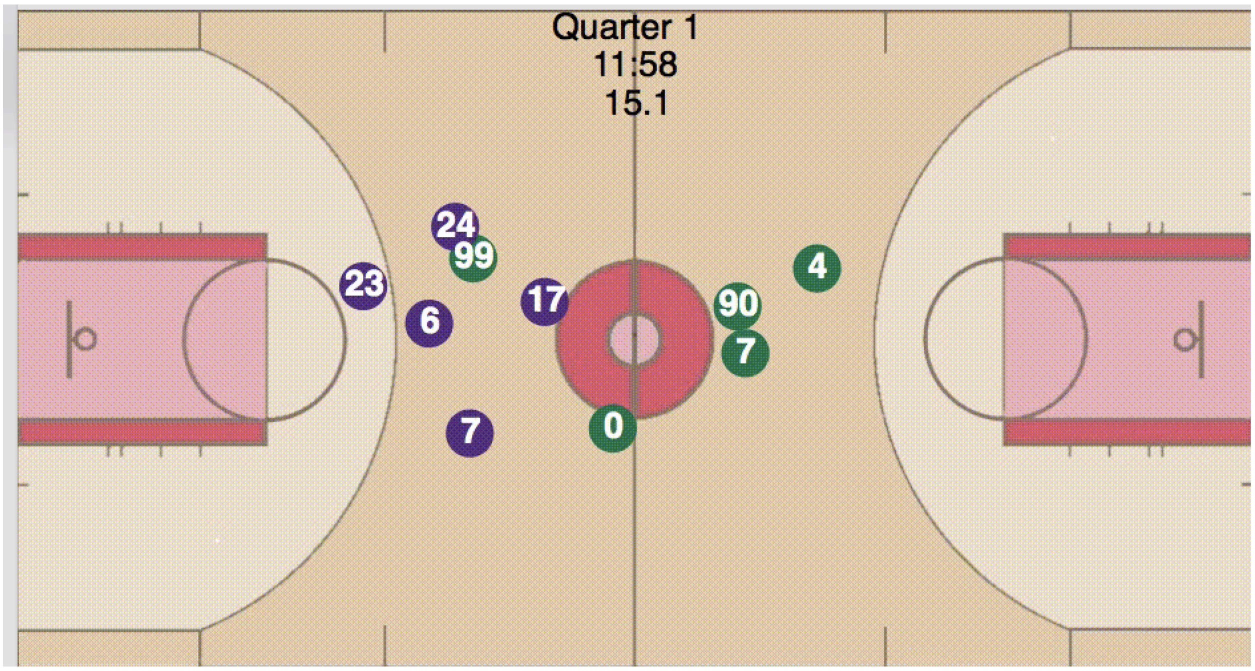
Arnav Jain, Darin Chaoui, Alex Kim

Introduction

PlayCaller.ai is a full stack application that takes raw image input of NBA basketball games and classifies the offensive and defensive favorabilities of the play. Built using react.js and python, the backend of PlayCaller.ai uses a unique image processing algorithm to make the raw game input digestible for the Convolutional Neural Network that classifies the images. We built both the backend and frontend over the month-long period of our 2023 winter break in addition to generating the dataset ourselves. The following sections will detail the quantifiers used in generating the dataset and also how the overall pipeline works itself.

Dataset

Generally speaking, machine learning with computer vision usually requires either a lot of complex data and a lot of computing power or less of both to achieve similar results. The first problem we encountered was how we were going to train our model at all without paying for cloud computing and working with those potential issues. Our solution came in the form of simple images and a smaller model—both of which would demand less data. To accomplish this we scraped individual JPG files from GIFs in the NBA-Player-Movements repository.

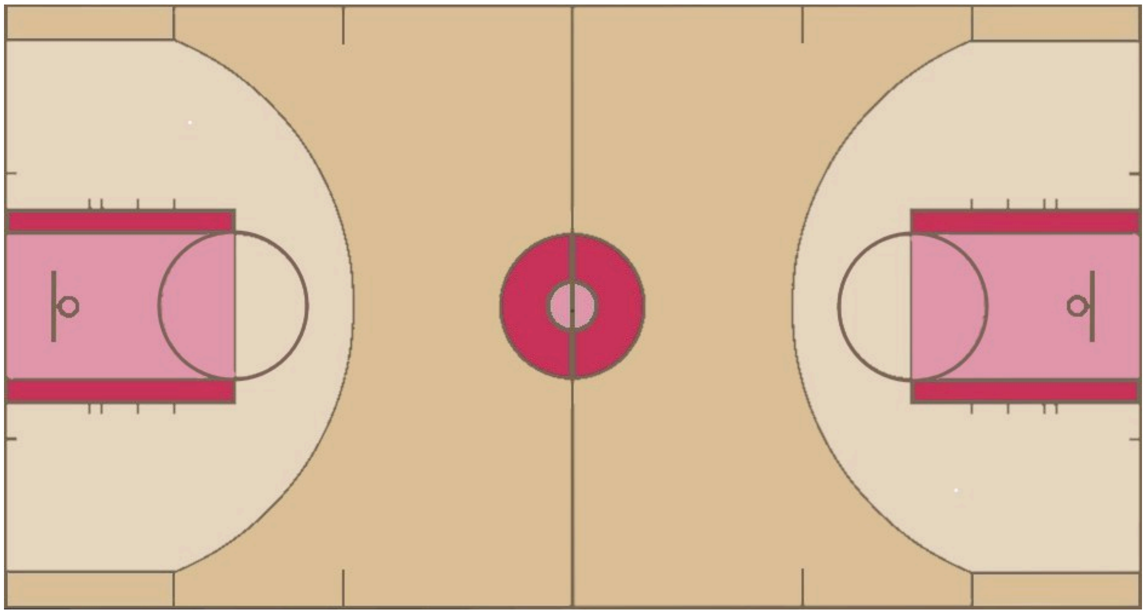


These images were sorted and labeled as either offensive (1) or defensive (0) and stored in a CSV file for training.

To standardize the categorization of the training data, we considered spacing, shot selection, isolation and rebounding mismatches for offensive favorability. On defense, we considered matchups, trapping, shot difficulty and transition defense. The resulting data was split 56.78% favoring the offense. The most important criteria on both ends of the floor was spacing and rebounding mismatches. Analytically good spacing indicates the threat of three pointers which is a favorable shot on the right team so we also assumed the coaches were making the right decisions. Offensive rebounding positioning and mismatches was also an important factor given that any extra possession is favorable and also an offensive rebound could lead to an easy putback in a lot of instances. We considered isolation plays also because a designed isolation demonstrates that the player is probably matched up well and the team/coaches think they have a good chance to score. In most cases, these factors were just flipped on defense but shot difficulty was more heavily weighted since that is often more affected by the individual defenders.

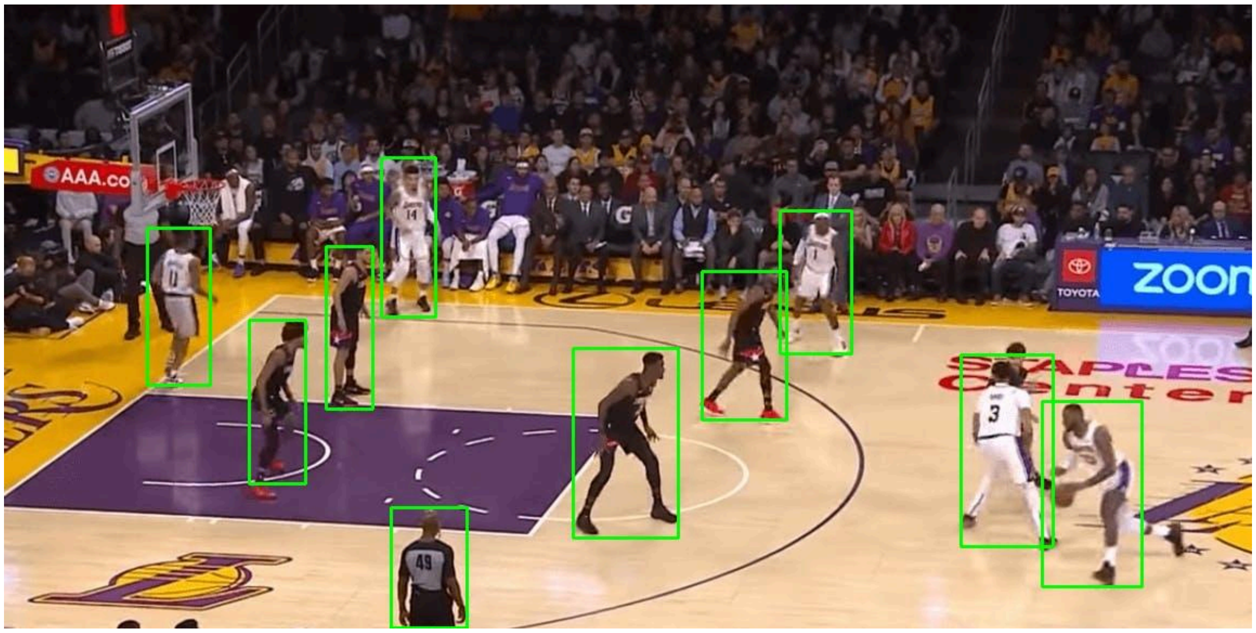
ML Pipeline

The pipeline we used to process and then classify raw footage was a multi step process that involved player detection and court mapping using distinct court features(the paint) to plot them on the standard 2 dimensional court.

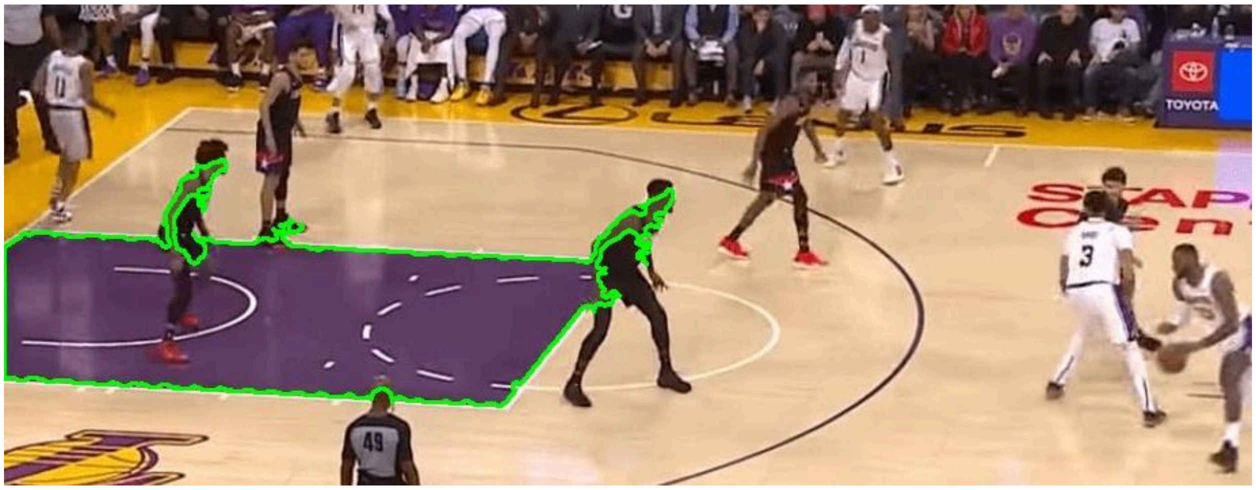


The image processing pipeline begins with player detection. We used the YOLOv4 model trained on the COCO dataset as our object detection model. The model would often detect elements in the crowd or at the broadcasting booth, thus we wrote a cropping function to eliminate as many outside objects as reasonable for the paint detection function.

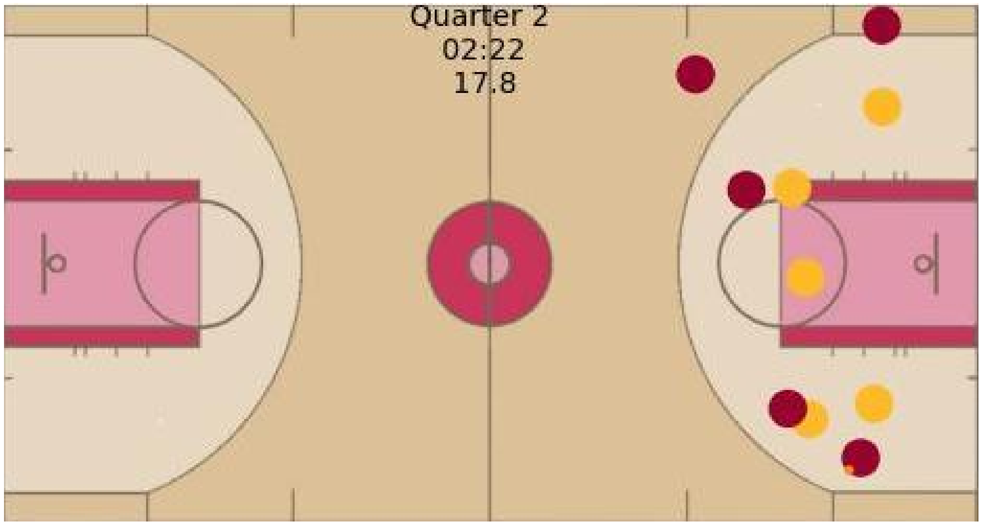
Then the YOLOv4 would detect the players in the image and return both their individual images and coordinates.



Using the cropped image, the find_paint_by_color_concentration function would detect the paint based on how different it is from the rest of the court. The diagonal of the paint is calculated and then used to find a midpoint which is translated to find a more accurate point underneath the basket. Using the player coordinates from the previous functions, vectors are cast from the basket to each player and then rotated depending on the angular displacement between the midpoint and basket.



Using a scale factor of the calculated length of the paint in the raw image and length of the paint in the standard court image, player positions are adjusted and assigned a “home” (red) or “away”(yellow) color.



This image would be input to the fine-tuned ResNet18 model which delivers a percentage of offensive and defensive favorability. This result is then displayed on the web page as the final result.