HOOPLAB: A CROSS-PLATFORM YOLO-POWERED MOBILE APPLICATION FOR REAL-TIME BASKETBALL SHOT DETECTION AND TRAJECTORY ANALYSIS

Derek Li [1], Austin Amakye Ansah [2]

[1] The Overlake School, 20301 NE 108th St, Redmond, WA 98053
[2] California State Polytechnic University, Pomona, CA 91768

ABSTRACT

Traditional basketball training relies on feedback from coaches and video playback, which can be delayed and inconvenient for solo practice. This paper introduces HoopLab, a cross-platform mobile application designed to provide immediate, comprehensive analysis of basketball shots. The app, built with Flutter, utilizes a custom-trained YOLO object detection model to analyze user-submitted videos. It features two distinct analysis modes: a backboard view that confirms if a shot is successful by tracking the ball's path through the rim, and a side-view that evaluates a shot's quality by comparing its actual path to a computationally generated optimal trajectory. The underlying YOLOv11 model was trained on a dataset of over 8,200 images, achieving a mean average precision (mAP@0.5) of 94.72% and demonstrating high performance in identifying key objects like the ball, rim, and player[10]. HoopLab offers players a powerful tool for instant, data-driven feedback to accelerate skill improvement.

KEYWORDS

Computer vision, AI, mobile, basketball, Flutter

1. INTRODUCTION

Many people aim to improve their skills in basketball, either for fun, academic merit, or sport. The current approach to improving is receiving feedback from teammates and coaches, and watching playback videos of the shots. The problem with this is that it requires constant communication with people who may not be available at the time. During private practice sessions, it may be worthwhile to receive instant, comprehensive visual feedback on basketball shots without having to wait for a response that may take a day to receive, depending on the other party's schedule.

We propose a cross-platform IOS and Android app that can be used to analyze basketball shots quickly and provide comprehensive feedback to steer improvement [11]. The app features a method selector that can be used to either record a new video or choose an existing video from the gallery. There is a video trimmer so that the user can focus on just the shot. The user has the option of choosing between a backboard mode and a side view mode. If the backboard is visible, the backboard mode takes whether the shot went in or not into account by determining whether the ball passed through the rim, around the middle of the backboard width. The side-view mode uses a different metric because determining whether the shot went in or not is impossible from

David C. Wyld et al. (Eds): DPPR, CNDC, AIAA, DSA, WeST, ACITY, IoTE – 2025
pp. 01-12, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.152402

one side with only detection models [13]. In the side-view mode, the shot is rated based on how close it was to the optimal trajectory. There is a timeline to move accurately across the video, to see how the data changes with time. If multiple shots are detected in one video, the user is given the option to move between them quickly without having to scroll through the video.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. YOLO Model Format Compatibility Challenges

One of the earliest and most complex challenges involved model format compatibility between YOLO, CoreML, and ML Package formats. The ML Package format stores model weights and metadata across multiple files, which introduced difficulties when integrating the model into the mobile deployment pipeline. In contrast, the CoreML (.mlmodel) format consolidates all required parameters into a single file, simplifying model loading and runtime execution.

During testing, the CoreML format demonstrated significantly better compatibility with the ultralytics_yolo package used within the Flutter environment. This improved compatibility reduced inference errors and simplified the deployment process, ultimately leading to a more stable and efficient model integration. As a result, CoreML was selected as the primary model format for this project.

2.2. Video Playback and Custom Slider Sync Issues

The HoopLab platform includes a custom-built video seeking toolbar, which introduced synchronization issues between video playback and the custom slider. After extended playback, the video would sometimes pause or resume unpredictably, and in some cases would stop playing entirely. Additionally, seeking through the video could cause the player to freeze. Resolving these issues required careful coordination between the video player state and the custom slider logic.

2.3. Low-Quality Thumbnail Frame Extraction

Early implementations of frame extraction relied on generating thumbnail images from video timestamps. While this method was simple to implement, it proved inefficient and produced low-resolution images. Since YOLO-based object detection requires frames with sufficient visual clarity to accurately identify objects, the low-quality thumbnails significantly reduced detection performance.

To address this limitation, FFmpeg was integrated into the workflow to extract high-quality frames directly from the video stream. This approach improved both the resolution and clarity of extracted frames, enabling more reliable object detection and significantly improving overall inference accuracy. Additionally, FFmpeg provided faster and more consistent frame extraction, making it better suited for real-time or near–real-time analysis.

3. SOLUTION

The user can either load their video clip through the camera or the gallery. Either method works because the video trimmer is source agnostic. The video can be from either the gallery or the camera. After the video is trimmed to leave only the shots of focus, the user has the option of

specifying whether they want to analyze the clip with the backboard in mind, or the camera side angle in mind.

When the backboard is taken into account, the metric used to determine the shot accuracy changes. The first method for determining whether a shot was taken was to see if the ball went into the backboard and hoop successfully, but this was not a good metric for determining shot accuracy in a sideways view, where it cannot be accurately determined whether the shot went in or not. To solve this problem, we created two analysis modes.

The updated approach leverages pose detection to identify video frames in which the user is performing a shooting motion. Once such a motion is detected, a new shot object is initialized at the corresponding timestamp and continues tracking until the ball enters the hoop. This method enables reliable shot object creation even in clips where the backboard is not fully visible.After tracking the actual ball trajectory, parabolic interpolation is applied to compute the optimal ball path. This is achieved by using the initial shot position and the detected hoop position to generate a smooth arc representing the ideal trajectory of the shot [14].

To provide the user with feedback, we compare the optimal shot path with the actual shot path to determine what changes in the vertical and horizontal need to be made to the actual shot path for it to be perfect. If the shot is similar to the optimal path, the feedback system lets the user know that they were close [12]. The system also tells the user how many pixels to the left or right the shot should've been taken in, or how many pixels off center the ball is.

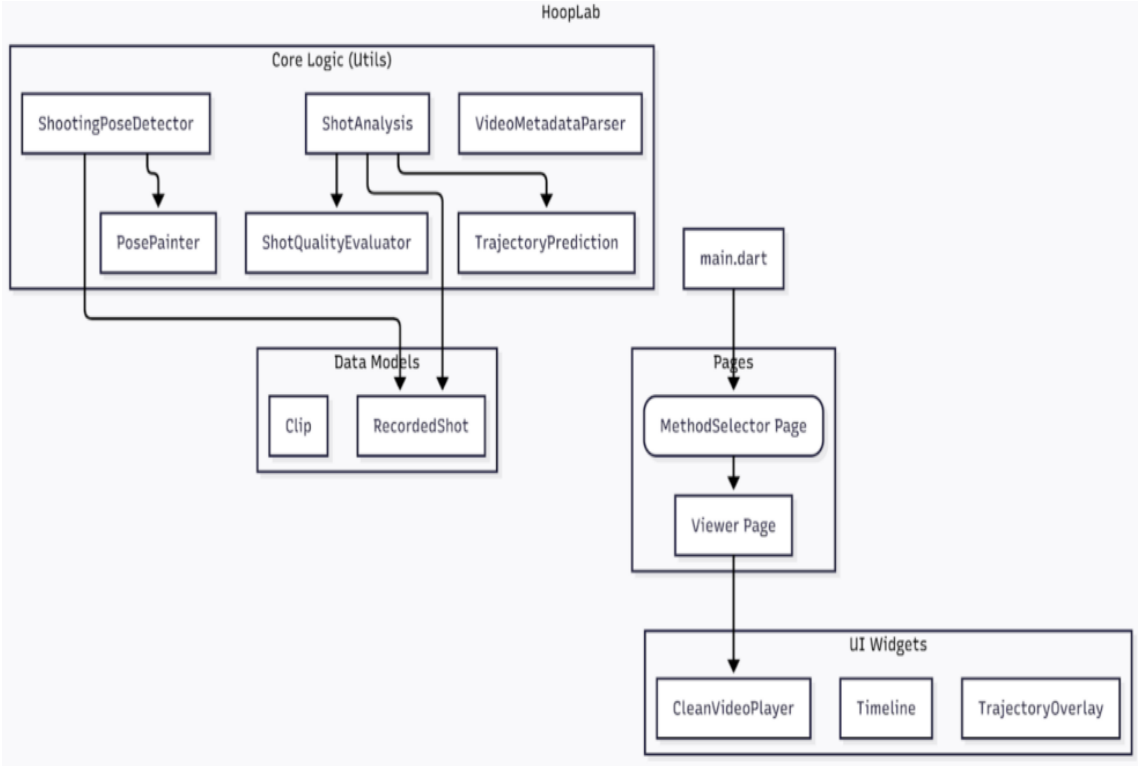

Figure 1. Overview of the solution

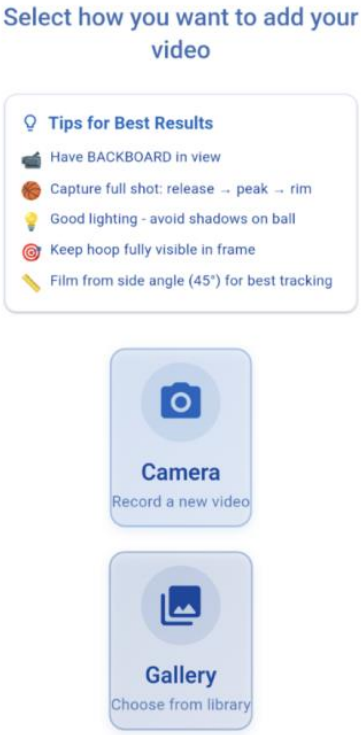

Figure 2. Screenshot of add video page

```
children: [
  Expanded(child: _buildLiveDetectionButton(theme)),
  const SizedBox(width: _buttonSpacing),
  Expanded(child: _buildCameraButton(theme)),
  const SizedBox(width: _buttonSpacing),
  Expanded(child: _buildGalleryButton(theme)),
],
);
```

Figure 3. Screenshot of code 1

The user has the option of using the gallery or recording a fresh new video. After the video file is recorded or selected, the app displays a shot trimmer, which allows the user to keep only the most important part of the shot. The trimmed clip is then saved to the filesystem, and its path is passed to the shot viewer page, which provides filesystem context for the video.The viewer page is where the user can analyze their shot and see where their shots may have gone wrong. The page features multiple detection modes to account for multiple court scenarios. For instance, one more uses a different algorithm to calculate shot accuracy when the backboard is in view, as opposed to the other, which accounts for the fact that the backboard may not be visible [15].

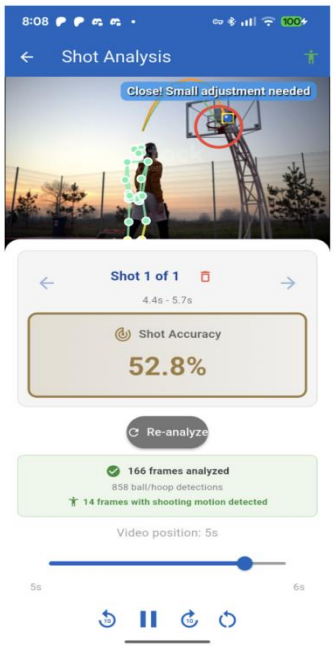

Figure 4. Screenshot of shot analysis page

```
// make sure that the number of points in ballPoints matches predicted trajectory, then
var Alength = predictedPath.length;
var Blength = ballPoints.length;

// In most cases, the predictedPath will have more points since it's required to be smoo
// so we minimize points up to where closest distance to hoop is
var x = (Alength / Blength);
var newASize = (Alength * x).round();

var newPredictedPath = predictedPath.sublist(0, newASize);
// cut ball points up to where closest distance to hoop is
var newBallPoints = ballPoints.sublist(0, newASize);

// calculate accuracy
double accuracy = 0.0;
for (int i = 0; i < newASize; i++) {
  final ballPoint = newBallPoints[i];
  final predictedPoint = newPredictedPath[i];
  final distance = (ballPoint - predictedPoint).distance;
  accuracy += distance;
}
accuracy /= newASize;

return accuracy;
```

Figure 5. Screenshot of code 2

The shot accuracy is calculated in two different ways depending on which analysis mode was selected.In this figure, we calculate the trajectory by using the distances between the actual ball positions and the predicted trajectory. The closer the points are to the optimal path, the higher the accuracy. If the ball misses or there's even a slight change in the difference, the accuracy is significantly affected.The shot accuracy is calculated in two different ways depending on which analysis mode was selected.

In this figure, we calculate the trajectory by using the distances between the actual ball positions and the predicted trajectory [4]. The closer the points are to the optimal path, the higher the accuracy. If the ball misses or there's a slight change in the arc difference, the accuracy is significantly affected.

Trajectory Prediction Method:

First, the system identifies two critical points: the last position above the rim and the first position below the rim. Using these points, a linear trajectory is fitted using least squares regression:

$$y = mx + b$$

Where m is the slope and $b$is the y-intercept computed from the coordinates of these two points.

The rim height is calculated as:

$$y_{\text{rim}} = y_{\text{hoop}} - 0.5 \times h_{\text{hoop}}$$

Where y_{hoop} is the center y-coordinate of the hoop and h_{hoop} is the height of the hoop bounding box.

Using the trajectory equation, the predicted x-coordinate where the ball crosses the rim plane is:

$$x_{\text{predicted}} = \frac{y_{\text{rim}} - b}{m}$$

The rim boundaries are defined as 80% of the hoop width:

$$x_{\text{rim\_left}} = x_{\text{hoop}} - 0.4 \times w_{\text{hoop}}$$
$$x_{\text{rim\_right}} = x_{\text{hoop}} + 0.4 \times w_{\text{hoop}}$$

A shot is counted as a make if the predicted trajectory passes through the rim:

$$x_{\text{rim\_left}} < x_{\text{predicted}} < x_{\text{rim\_right}}$$

Alternatively, if the ball enters the rebound zone (defined with a 10-pixel buffer):

$$x_{\text{rim\_left}} - 10 < x_{\text{predicted}} < x_{\text{rim\_right}} + 10$$

The shot may still be counted as a make to account for close attempts that bank off the backboard or rim. For the sake of shot improvement, whether the ball goes in or not is trivial, but it is a decent enough metric to determine whether the shot was good or not.

Corrected Arc Trajectory Generation

The corrected arc shows what trajectory the ball should have followed to make the shot. This is created using parametric equations where a time parameter t ranges from 0 to 1, representing the progression from the shot's starting point to the hoop [5].

The arc is divided into n steps, where n is the number of interpolation points from the start to the end of the arc.

If the x and y displacements, starting ball position, hoop position, and the peak height are known, a parametric formula can be used to calculate the optimal parabolic path from the start of the shot to the hoop.

$$x = x_{\text{start}} + dx \times t$$

$$y = y_{\text{start}} + dy \times t + 4 \times (y_{\text{peak}} - y_{\text{start}}) \times t \times (1 - t)$$

Figure 6. Result when N = 2

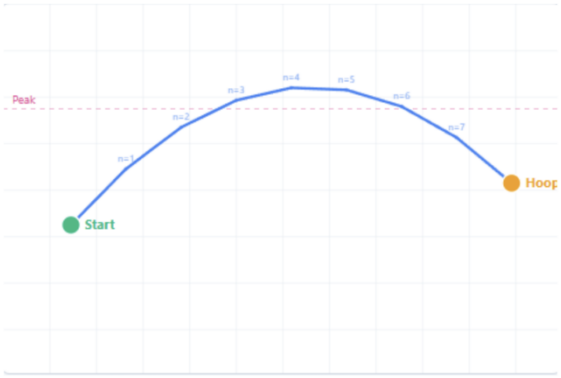

Figure 7. Result when N = 8

Hooplab uses an n value of 30 to maintain arc smoothness at very little cost. When the hoop position changes, the app can quickly recalculate the arc using Flutter's CustomPaint API [8].

Figure 8. Low shot accuracy

We also take into account whether the ball made a full movement into up and down regions of the hoop and check to see if the rim bounding box was crossed.

The shot accuracy in this simulation is determined by how close to the center of the hoop the ball was.

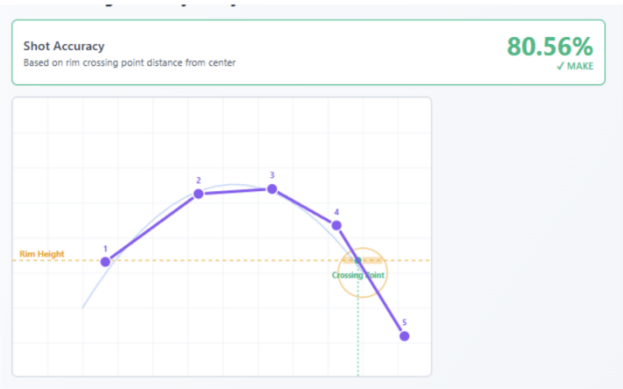

Figure 9. High shot accuracy

Video Trimmer

To prevent exceedingly long videos from being processed, Hooplab features a widget that lets users crop videos to acceptable lengths. Since the frame extraction relies on FFmpeg, and the YOLO inference is done rapidly on-device, it does not take more than 2 minutes to process a relatively long video [6]. For the best experience, a single basketball shot clip of around three to six seconds is recommended.

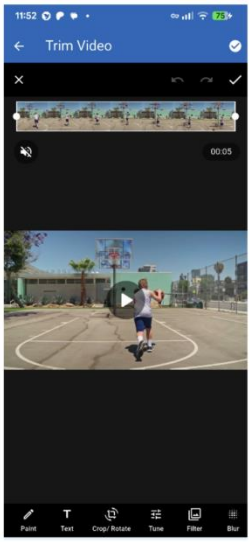

Figure10. Screenshot of trim video

```dart
if (trimResult != null && mounted) {
  // Generate trimmed video
  final trimmedPath = await _generateTrimmedVideo(
    videoPath,
    trimResult,
  );

  if (trimmedPath != null && mounted) {
    // Navigate to viewer with trimmed video
    await Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => viewer.ViewerPage(videoPath: trimmed
      ),
    );
  }
}
```

Figure 11. Screenshot of code 3

We utilized a package called ProVideoEditor to trim the videos programmatically [7]. The package contains a video trimming user interface and generates thumbnails for visual feedback.

After the video is trimmed, it is exported to a temporary location, then passed as a path to the viewer page.

4. EXPERIMENT

There was a need for a model that could perform multi-class detection for rims, hoops, balls, and players simultaneously in a single frame, with a very high inference rate. YOLO was the model of choice for this, as it is very easy to train and has mobile APIs ready for use.

The dataset consisted of 8217 balls, rims, made shots, hoops, null images, and people. The model was trained locally on an Nvidia RTX 3060 for 1 hour over 31 epochs.

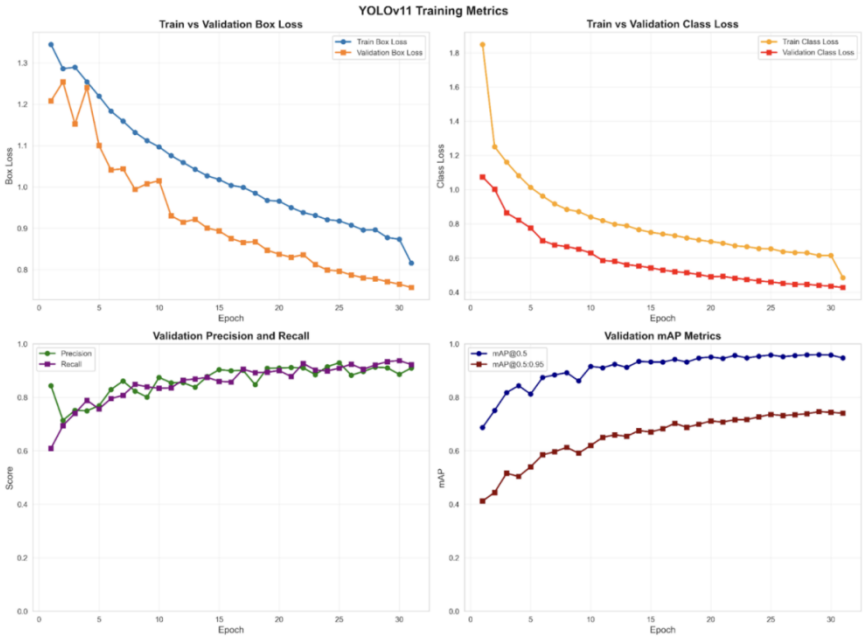

Figure 12. YOLOv11 training metrics

The model trained locally performed better than the one trained on the cloud platform. mAP@50 and mAP@50:95 for the validation set trained locally were 94.72% and 74.06% respectively.

mAP@50 and mAP@50:95 for the validation set trained online were 91.7% and 67.46% respectively.

The precision on the local model was 90%, and on the cloud-trained model, also 90%.
Recall on the local model 92% whereas the cloud model had a recall of 84%

The tables below summarize the results of the local training process.

Metric	Type	Initial Value (Epoch 1)	Final Value (Epoch 31)
Box Loss	Train	1.34468	0.81575
Box Loss	Val	1.20808	0.75678
Class Loss	Train	1.84907	0.48548
Class Loss	Val	1.07448	0.42780

Metric	Final Value (Epoch 31)	Max Value
Precision	0.90990	0.92946
Recall	0.92257	0.93756
mAP@0.5	0.94723	0.95972
mAP@0.5:0.95	0.74055	0.74645

Figure 13. Table of experiment

Although the training and validation box losses were quite high, the model performed well visually and in the app for all of its applications. This is due to the fact that it was trained from a pre-trained backbone provided by the Ultralytics team.

Figure 14. Figure of posture catch

Cloud Model Training

We trained the model again on the cloud using the same dataset on 41 epochs to see if we could get a more performant model.Similar to the first training process, we used 8217 images to train the model, starting from a yolov11 backbone.

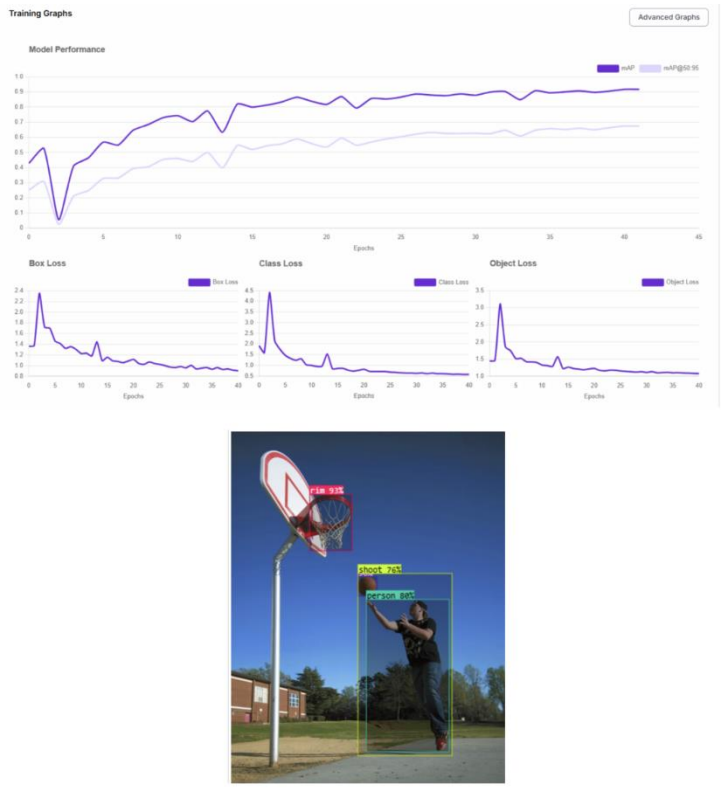

Figure 15. Training graphs

The model trained on the cloud platform over 41 epochs achieved strong, albeit slightly lower, performance compared to the locally trained model. The final mean average precision at an IoU of 0.5 (mAP@0.5) stabilized around 91.7%, while the mAP@0.5:0.95 reached 67.46%. These results are marginally lower than the 94.72% mAP@0.5 and 74.06% mAP@0.5:0.95 achieved by the local model.

The training graphs demonstrate a stable learning process. The Box, Class, and Object Loss metrics show a rapid decrease in the initial epochs before converging and flattening, which indicates that the model effectively learned to identify, classify, and locate objects within the dataset.

A detailed breakdown of the average precision by class reveals excellent performance on key object types. The model was exceptionally accurate at identifying the 'rim' (99.8% mAP) and 'person' (94.8% mAP). Performance for 'shoot' (91.8%), 'ball' (88.0%), and 'made' (88.0%) was also robust, contributing to an overall validation mAP of 92.0% across all classes. While the local model proved superior, the cloud-trained model is still highly effective and performs its required tasks reliably within the application.

5. RELATED WORK

This paper proposes using a Recurrent Neural Network to determine whether a shot is successful or not by utilizing only the ball's positional information [1]. Their goal was to determine whether a shot would be successful from two to eight feet from the basket. The dataset consists of 20,000 three-point shots from 631 games. Compared to traditional approaches, the RNN approach achieved an AUC performance value of 0.843, compared to 0.558 and 0.719 for a general linear model and gradient boosted machines. The performance of the neural network increases the shorter the distance of the ball to the center of the rim decreases.

The goal of the paper is to accurately detect basketball shooting motions by utilizing a sequence of video frames as opposed to a single frame of video [2]. Their algorithm works by normalizing the input frames to remove unnecessary noise like backgrounds and lighting. If the posture of the shooter is incorrect, a pix2pix GAN is utilized to correct this. Using the shot posture, it's able to vote on whether the shot will be a make or miss.

The article presents a method for predicting fixed-point basketball shot success by focusing on the player's biomechanics [3]. Their approach uses human pose estimation to extract key body angles—such as those of the elbows, shoulders, and knees—during the shooting motion. This kinematic data is then fed into a support vector machine (SVM) to classify the shot as a "hit" or "miss." Unlike HoopLab's focus on the ball's trajectory, this method evaluates the quality of the shot based entirely on the shooter's form. It provides a different dimension of feedback by analyzing the cause (the shooting posture) rather than the effect (the ball's flight path).

6. CONCLUSIONS

There are a few limitations to the current approach, being that because the ball is tracked directly, and the assumption is that the backboard and full shot motions are in view, videos where the camera is behind the hoop cannot be used.

When multiple players appear in the same video frame, the current algorithm may struggle to accurately distinguish the shooter from other players on the court. In addition, if the confidence threshold for object detection is set too low, non-basketball objects may be incorrectly identified as basketballs.The current system also does not explicitly account for depth information. As a result, a basketball moving farther away from the camera may not be correctly recognized as a valid shot. Incorporating depth-aware analysis could improve detection accuracy in such scenarios.Future improvements could leverage existing research on accurately identifying shooting poses to better isolate the shooter [9]. Additionally, the model could be retrained to classify players who are holding the ball, enabling more reliable differentiation between the shooter and other players in the frame.

REFERENCES

[1] Shah, Rajiv, and Rob Romijnders. "Applying deep learning to basketball trajectories." arXiv preprint arXiv:1608.03793 (2016).
[2] Egi, Yunus. "Basketball self training shooting posture recognition and trajectory estimation using computer vision and Kalman filter." Journal of Electrical Engineering 73.1 (2022): 19-27.
[3] Liang, Zeyu, et al. "Basketball detection based on YOLOv8." PLoS One 20.8 (2025): e0326964.
[4] Yoon, Young, et al. "Analyzing basketball movements and pass relationships using realtime object tracking techniques based on deep learning." IEEE Access 7 (2019): 56564-56576.
[5] Zhu, Luyang, et al. "Reconstructing nba players." European conference on computer vision. Cham: Springer International Publishing, 2020.

[6] Mehrasa, Nazanin, et al. "Deep learning of player trajectory representations for team activity analysis." 11th mitsloan sports analytics conference. Vol. 2. 2018.

[7] Pascual, Alejandro Rodriguez, et al. "Understanding why shooters shoot--An AI-powered engine for basketball performance profiling." arXiv preprint arXiv:2303.09715 (2023).

[8] Pan, Yulu, Ce Zhang, and GedasBertasius. "BASKET: A Large-Scale Video Dataset for Fine-Grained Skill Estimation." Proceedings of the Computer Vision and Pattern Recognition Conference. 2025.

[9] Chen, Chien-Chang, et al. "Video based basketball shooting prediction and pose suggestion system." Multimedia Tools and Applications 82.18 (2023): 27551-27570.

[10] Khanam, Rahima, and Muhammad Hussain. "Yolov11: An overview of the key architectural enhancements." arXiv preprint arXiv:2410.17725 (2024).

[11] Hui, Ng Moon, et al. "Cross-platform mobile applications for android and iOS." 6th Joint IFIP Wireless and Mobile Networking Conference (WMNC). IEEE, 2013.

[12] Tadelis, Steven. "Reputation and feedback systems in online platform markets." Annual review of economics 8.1 (2016): 321-340.

[13] Zaidi, Syed Sahil Abbas, et al. "A survey of modern deep learning based object detection models." Digital Signal Processing 126 (2022): 103514.

[14] Snyder, Willard M. "Continuous parabolic interpolation." Journal of the Hydraulics Division 87.4 (1961): 99-111.

[15] Wulf, Gabriele, and Jiang Su. "An external focus of attention enhances golf shot accuracy in beginners and experts." Research quarterly for exercise and sport 78.4 (2007): 384-389.