Interim Report

Gleb Bikushev

June 12, 2024

1 Introduction

1.1 Brief introduction to problem

In the fast-evolving landscape of sports analytics, the integration of data-driven decisions has become crucial to advancing athletic performance. This Master's dissertation introduces a web application that leverages data analytics to improve basketball training experience and effectiveness by providing players with detailed feedback on their shooting performance. Utilizing computer vision techniques, this platform provides analysis of training session videos uploaded by the users, offering immediate feedback on their shooting performance, providing detailed statistics on their shots, including successful attempts and misses. Further, users can track the results of their training sessions, visually observing how their throwing accuracy changes over time.

1.2 Research questions

Main question:

1. How can computer vision techniques be adapted and incorporated in a web application to specifically recognize and analyze users' basketball shooting activities?

Supporting questions:

1. To what extent can the accuracy and reliability of a YOLOv8 model be improved through training to detect and track player, ball and basketball hoop objects on a basketball court for a camera positioned at specific points on the court?
2. Can an accurate and reliable framework for detecting throws and successful shots in the video be developed based on specifically trained YOLOv8?
3. Can a web application be developed with an integrated algorithm for detecting throws and goals that allows users to upload a basketball video and get statistics of shots from it?
4. How does quantitative feedback affect the training habits and score performance improvements of basketball players?

1.3 Problem definition

The traditional approach to tracking sports performance, namely counting the number of shots and goals in basketball, often relies on subjective observation and manual data recording, which are inherently limited by human error and scalability issues. This project proposes a solution to automate the process using a computer vision system that not only captures every detail but also provides quantitative feedback that is both accurate and immediate.

1.4 Project relevance in the field of Data Analytics

This project contributes significantly to the field of data analytics by applying deep learning and computer vision to the domain of performance monitoring in basketball, enhancing the precision and efficiency of sports training analytics. By transforming raw video data into actionable insights utilizing

the YOLOv8 model and score detection algorithm, the project extends the application of data analytics to the basketball shots performance monitoring within training session. In addition, the decision of adding clear and concise visualization of the users score performance using the line graphs and bar charts makes the project relevant to the Data Analytics domain.

1.5 Core technology, architecture and research processes used

1.5.1 Data collection

The initial stage involved the collection of a diverse and robust dataset necessary for training an accurate YOLO model. This dataset includes various basketball game scenarios captured from YouTube Shorts videos, which provide a rich source of diverse environments and play styles. In order for the model to be able to identify the necessary objects on the basketball court from specific angles and specific height values, external participants were involved. They provided videos from their throwing practice sessions recorded with the phone located both on the ground and on some height using a tripod. Further, all the YouTube Shorts videos and videos from external participants were uploaded on the Roboflow, and frames were extracted at a frequency of around 10 frames per second to create a large dataset. Using the Roboflow labeling tool, each frame was thoroughly labeled to detect three objects on the basketball court: the player, the ball, and the basketball hoop.

1.5.2 YOLO Model training

The core of the technical implementation involves training a custom YOLOv8 model. This stage was approached by first using a large pre-trained YOLOv8 model from Ultralytics as a base to benefit from its existing learned features, which were then fine-tuned with our custom dataset through 100 training epochs. This approach ensured that the model not only learned the general characteristics of the objects of interest but also adapted to the specific nuances present in basketball training videos.

1.5.3 Throws and goals detection algorithms

Post-training, the focus shifted to developing a sophisticated algorithm capable of interpreting the video data to determine the number of shots, and distinguishing between successful shots and misses. Having the model that accurately detects all required objects on the basketball court, the software approach for throws and goals detection was developed, which is based on the coordinates of bounding boxes of a player, ball and basket from each video frame and the interactions between these objects and and special rectangular zones built near the basketball hoop to determine the presence of the ball in them.

1.5.4 Web application development

The final stage of the project involves creating a cloud-based web application. This app will allow athletes to upload training videos and receive detailed performance statistics. It will be powered by the FastAPI Python framework, processing videos and managing user data within a PostgreSQL database. SQLAlchemy will bridge Python and the database, allowing the use of Python classes over raw SQL. This approach supports heavy video processing demands and ensures secure and efficient data management.

1.6 Hypothesis for a solution

The hypothesis behind this study is that an accessible, easy-to-use web application powered by a custom-trained YOLOv8 model will provide accurate analytics on basketball shots. This system is expected to significantly improve the training quality by offering immediate feedback of player throwing accuracy, allowing players to clearly track their progress in improving their throwing skills. The camera setup's simple design, which requires minimal installation and utilizes common mobile devices, is anticipated to increase the adoption rate among basketball players of varying skill levels.

1.7 Structure of the report

The structure of the report is as follows:

1. Introduction
    (a) Brief introduction to problem
    (b) Research questions
    (c) Problem definition
    (d) Project relevance in the field of Data Analytics
    (e) Core technology, architecture and research processes used
        i. Data collection
        ii. YOLO model training
        iii. Throws and goals detection framework
        iv. Web application development
    (f) Hypothesis for a solution
    (g) Structure of the report
2. Literature Review
    (a) Computer vision in different domains
    (b) Computer vision in sports analytics
    (c) Computer vision in basketball
    (d) OpenPose
    (e) YOLO
    (f) ByteTrack
    (g) Practical applications of basketball analytics
    (h) Conclusion
3. Exploration of Data and Methods
    (a) Dataset creation
        i. Data Collection
        ii. Data Labelling
        iii. Data Splitting
        iv. Data Preprocessing and Augmentation
    (b) Model Training and Evaluation
        i. Model Training
        ii. Model Evaluation
        iii. Model Inference
    (c) Throws and goals detection algorithm
        i. Building rectangular zones
        ii. The logic of the algorithm
        iii. Video Requirements
4. Proposed Future Analysis
    (a) Applying ByteTrack
    (b) Evaluation of throwing and scoring algorithm
    (c) Web application development
    (d) Future work plan

2 Literature Review

2.1 Computer vision in different domains

Due to the rapid development of artificial intelligence technologies, specific areas including computer vision and computer image processing, have introduced innovations across a broad array of disciplines. Computer vision is a crucial aspect of artificial intelligence, aiming to empower computers in comprehending and interpreting visual data from images or videos [29]. It involves object classification, detection and segmentation. The following studies demonstrate the use of computer vision in various fields.

The paper [32] showcases how computer vision techniques can be applied to the study of animal ecology. Describing in detail the use of computer vision for observing, counting and identifying species in their natural habitat, it highlights the enormous potential of computer vision, bridging the gap between technology and ecological research.

Another example of implementing computer vision in different domains is paper [2]. By demonstrating how computer vision can enhance security systems through automatic face detection and movement recognition, this paper contributes valuable insights into practical applications of computer vision technologies.

2.2 Computer vision in sports analytics

In the field of sports and sports analytics, computer vision has also made a great contribution, offering considerable and in-depth insights that previously can not be obtained. This detailed level of analysis, that computer vision offers in the realm of sports analytics, helps coaches develop strategic game plans, helps players improve their skills by providing feedback on their moves [31], and increases audience engagement through advanced statistics and visualization [10].

For instance, the paper [26] discusses current commercial applications that are using computer vision for sports analysis. The authors consider team sports such as football (soccer), basketball, and American football, where player and ball tracking are crucial for tactical analysis and coaching. Also they involved racket sports including tennis and badminton, where ball tracking systems are used for officiating and enhancing broadcast coverage. In the context of individual sports, the authors mentioned motion capture for training professional athletes, where precise movements are analyzed for performance improvement.

The paper [19] provides a comprehensive review of computer vision techniques in sports (specifically in soccer, basketball, cricket, and badminton) by discussing and summarizing the variety of published works about application-specific sports-related problems. Broadly speaking, these studies focus on tasks like player detection and tracking, balls and players trajectory prediction in real-world sports scenarios, as well as identifying the tactics used by the team and categorizing different sporting events. For example, having the problem of detecting and tracking a player and a ball in soccer, the proposed methodology was using YOLOv3 and SORT algorithms (will be further discussed in section "YOLO"). This approach achieved a high tracking accuracy of 93.7% across various object tracking accuracy metrics. It operated at a detection speed of 23.7 frames per second (FPS) and a tracking speed of 11.3 FPS. Despite its effectiveness in handling partial occlusions and tracking players and the ball as they reappear after being momentarily out of frame, the methodology struggled with significant occlusions, where it failed to maintain accurate tracking. Additionally, the approach of combining such techniques as YOLO and Joy2019 was used to track ball movements and classify players in basketball games with the personal numbers on their jerseys. It achieved a precision of 74.3% for jersey number recognition and a recall of 89.8% for player recognition. However, the system faced issues when players overlapped, since YOLO mistakenly identified overlapping players as a single entity.

2.3 Computer vision in basketball

In basketball analytics, computer vision approaches have been used for ball trajectory analysis, players motion capture, object detection (player, ball, basketball hoop) and event categorization. [7]. Event categorization refers to the process of classifying various actions and occurrences during a basketball game into distinct categories, such as shots, passes, and fouls, using computer vision techniques. Better training techniques and tactical insights for teams and coaches have been made possible by the greater

understanding of the game and player performance that has resulted from advances in computer vision and artificial intelligence, in general.

For example, in the article [15] the authors utilized a position tracking algorithm based on action division in the field of artificial intelligence to determine the position of the basketball and identify classified actions of basketball players. These actions are decomposed into instantaneous and continuous operations, generating movements of upper (arm movements) and lower (leg movements) limb units based on limb angle changes in the movement process. The study reported high accuracy and recall rates, with the four different Machine Learning (ML) classification algorithms such as Decision Tree (DT), Naive Bayes (NB), Support Vector Machine (SVM) and Artificial Neural Network (ANN), demonstrating an average accuracy of movement recognition between 96.99% and 99.19% for lower limb movements, and between 84.89% and 92.19% for upper limb movements, including activities like dribbling, walking, running, and others. Therefore, this paper enables coaches to easily identify incorrect postures and allows for immediate correction, enhancing the training quality. However, the authors mentioned that there is no objective data to support the standards of basketball dribbling posture internationally. This suggests a future improvements of the research in the form of developing more standardized and objective criteria for evaluating basketball dribbling postures.

Another study [33] proposes a method that significantly improves the accuracy of basketball action recognition by addressing the limitations of the original C3D (Convolutional 3D) convolutional neural network, which struggled with focusing on keyframes and extracting sufficient feature information. The new method leads to an average accuracy of posture recognition of basketball players of more than 97%. Thus, by enabling precise control over athletes' sports postures, the authors significantly improve training effectiveness.

Based on the above-mentioned reasons, ball and basketball players detection and tracking in real time or through the analysis of provided video fragments have been the subject of different studies that have employed a variety of object detection techniques and algorithms.

2.4 OpenPose

One of the widely used techniques for basketball players detection and their pose estimation is **OpenPose** [20]. A special feature of this algorithm is that it recognizes multiple persons not just as bounding boxes; it is real-time human pose estimation algorithm, that capture 2D positions of a person's joints and skeleton wireframe of the body, and then draw lines between these keypoints, recreating person's exact pose (Figure 1).

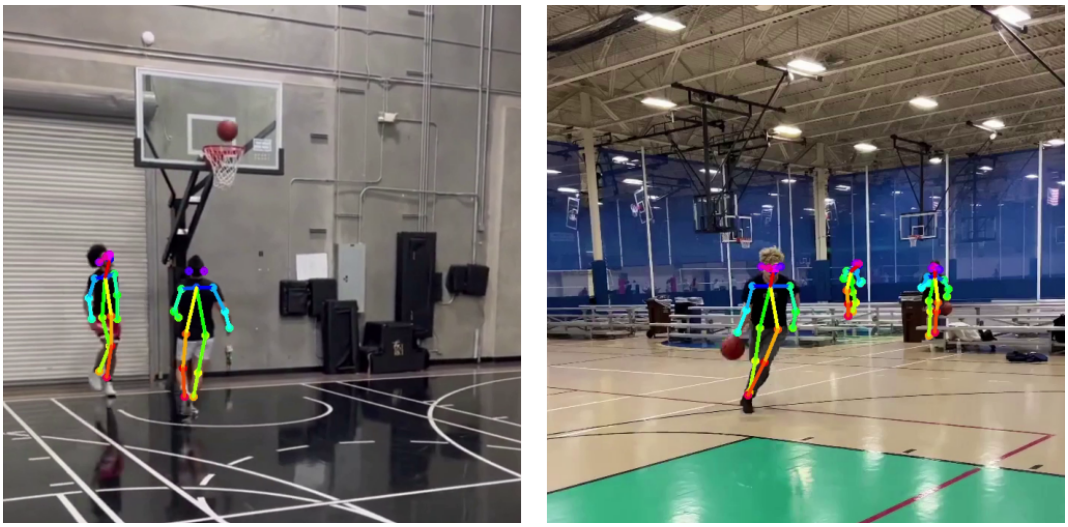

Figure 1: Examples of video frames processed by OpenPose

According to [5], the algorithm architecture is designed in such a way that the process of human pose estimation is divided into two branches that work concurrently: the first branch utilizes a feed-forward network to produce a series of 2D confidence maps that identify the locations of body parts of all individuals on the picture, and the second branch generates a series of 2D vector fields of body part

affinities, which represent the level of association between these parts. These confidence maps and affinity fields are then combined and analyzed through a greedy inference process, which ultimately outputs the 2D keypoints for every individual in the image.

Regarding the application of OpenPose technique in basketball, the study [6] can be a good example of that. The authors of this paper proposed a video-based basketball shooting prediction and posture suggestion system that leverages the OpenPose system for detecting human joints and, as a result, pose estimation, without the need for attaching sensors to the athletes. Concerning the experimental results, the authors evaluated the five curve similarity algorithms for predicting the basketball shooting outcomes based on the players postures for two different groups: general basketball class students and university team players. The authors reported that the highest accuracy rates were achieved when taking into account the overall differences in shooting postures among all players, rather than individually, and amounted to 87% for basketball class students dataset and 50% for university team players. Moreover, the study points out the potential for future improvements by incorporating more robust features like depth information or moving speed for better shooting outcomes predictions. Regarding the posture suggestion system, the results showed that it can effectively adjust incorrect poses and provide recommendations for players.

Additionally, the study [25] explores how OpenPose can help to avoid occlusion of human bodies and improve their detection efficiency, compared with using only color information, while detecting players on the basketball court. Specifically, the authors employed cameras placed at three different angles to capture comprehensive visual data, reducing the chances of player occlusion. The authors conclude that applying OpenPose with combination of these cameras clearly reduced the occlusion rate between basketball players and its use to identify players is very effective compared to using team uniform color alone. Particularly, traditional methods resulted in occlusion rates of 0.14% for three players, 0.26% for two players, and 0.42% for one player, with a 0.21% rate when no occlusion occurred. In contrast, the integration of OpenPose effectively reduced the occlusion rate to nearly zero, as it consistently allowed players to be detected from at least two different viewpoints, effectively avoiding occlusions in all tested instances. However, the effectiveness of the system relies heavily on the availability of multiple camera viewpoints to resolve occlusions. This could limit the method's applicability in environments where only one or two cameras are available.

2.5 YOLO

Another popular object detection algorithm in the field of computer vision, that is worth emphasizing, is **YOLO** (You Only Look Once). It marks objects in the picture using rectangular bounding boxes with corresponding class labelling and its confidence score (Figure 2).

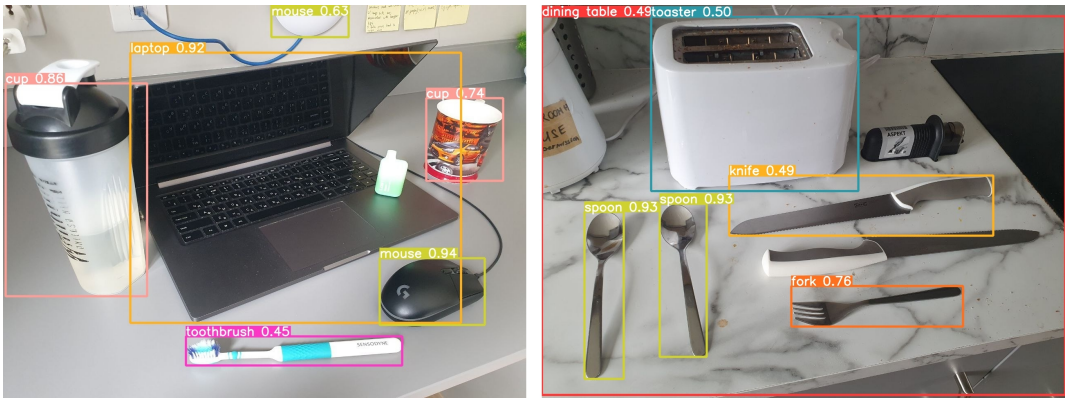

Figure 2: Examples of pictures processed by YOLOv8s model

The authors in the paper [23] first proposed this novel method. Generally speaking, contrary to other approaches, they frame detection as a single regression problem and, therefore, don't build a complex pipeline. More specifically about the YOLO method, the input image is first split into a grid where each cell predicts multiple bounding boxes and their confidence scores. Each bounding box includes the box center coordinates (x, y), its dimensions like height (h) and width (w), and a confidence score (c) that indicates the likelihood of an object being present and the accuracy of the

box. Simultaneously, each grid cell predicts conditional class probabilities ($\Pr(\text{Class}_i \mid \text{Object})$) based on the presence of an object. During detection, these probabilities are combined with the confidence scores to produce class-specific confidence scores (C) for each box, reflecting both the presence of a class within the box and the precision of the fit. Thus, each bounding box represent following vector:

$$\text{bounding box} = (x, y, h, w, C, \text{class})$$

where the categorical variable "class" reflects the specific class to which the object belongs. This method allows YOLO to predict multiple bounding boxes and class probabilities across the entire grid at once, simplifying object detection. This makes the proposed algorithm the fastest general-purpose object detection approach that stands out for its exceptional balance of speed and accuracy, compared to other classifier-based approaches. Moreover, since the model is supposed to be trained directly of full images, meaning it uses entire, unmodified images during its learning process, this method significantly simplified the task of training custom detection model. This reason has served as the impetus for many projects and studies in the field of object detection, particularly in such sports game with fast movements of the players and the ball as basketball.

Before discussing in detail some of the papers devoted to the topic of applying YOLO technology in basketball analytics, it is worth briefly mentioning the story of how YOLO has developed and improved over time:

- Introduced in 2015, YOLO quickly became popular for its impressive speed and precision.
- In 2016, YOLOv2 [21] was unveiled, enhancing the original with features like batch normalization, anchor boxes, and dimension clustering.
- YOLOv3 [22], released in 2018, advanced the architecture further by integrating a more robust backbone network, multiple anchor boxes, and spatial pyramid pooling.
- Launched in 2020, YOLOv4 [4] brought new features such as Mosaic data augmentation, an anchor-free detection head, and an innovative loss function.
- YOLOv5 [27] enhanced the model's capabilities with additions such as hyperparameter optimization, integrated experiment tracking, and automatic export to popular export formats
- In 2022, Meituan released YOLOv6 [16], which now powers many of its autonomous delivery robots.
- YOLOv7 [30] introduced additional capabilities like pose estimation for the COCO keypoints dataset.
- YOLOv8 [28] is the latest version of YOLO by Ultralytics, which continues to refine the series with significant advancements, enhancing performance, flexibility, and efficiency. YOLOv8 supports a broad array of vision AI tasks, such as detection, segmentation, pose estimation, tracking, and classification, making it adaptable to a wide range of applications and industries.

Let us now examine several studies regarding the application of the YOLO algorithm in basketball.

In the study [1], authors introduced a multi-detection and tracking framework capable of precisely detecting and tracking basketball players using only broadcast videos. This framework is based on YOLOv2, state-of-the-art real-time object detection system, and SORT [3], a simple but accurate object tracking algorithm. The framework was trained and evaluated on NCAA basketball dataset with 8787 images for training and 1000 images for testing set. The results demonstrates that the proposed approach achieved an Average Precision (AP) of 0.89 for the standard criteria (IOU0.5) and 0.63 for the harder criteria (IOU0.7). In other words, when using a threshold of 0.5 for the Intersection Over Union (a measure of overlap between the predicted bounding box and the ground truth), the model achieves an average precision of 89%, while using a IOU threshold of 0.7, a stricter criterion, the AP decreased to a value of 63%.

In [12] the authors consider multiple tracking of basketball players with basketball court monitoring,, aiming for good accuracy and real-time operational efficiency. The authors compare different baseline detectors such as Faster-RCNN [24] and YOLOv3 to assess their impact on tracking accuracy in the basketball court scenario. For this specific task, they developed a new basketball court dataset

that includes 15 labeled sequences of frames (3000 each) from 15 basketball videos with a duration of 2 minutes. Using various evaluation metrics, results demonstrates that Faster-RCNN surpasses YOLOv3 in terms of accuracy, but lacks in execution efficiency, since YOLOv3 achieves near real-time speed in players detection, while Faster-RCNN detection speed is just 3 frames per second.

The paper [34] focuses on automatically classifying players, tracking ball movements and analyzing pass relationships in basketball games using YOLO, as the core of the presented system. Pre-trained with Microsoft's COCO dataset [17], YOLO was trained on the 16,000 ground truth boxes of balls, players and jersey numbers from 1204 video frames. This made it possible to create a model that detects the above-mentioned objects with high efficiency. It classifies detected objects by identifying players and recognizing their jersey numbers, which is crucial for tracking individual player movements and ball possession across frames. Also, the study adapted YOLO to track players and the ball even when players move out of the frame or overlap each other. This adaptation involved using contextual information from previous or subsequent frames to maintain continuity of player tracking despite difficult conditions such as rapid camera shifts.

The study [11] went a step further and used YOLO for a camera-based basketball scoring detection method. The YOLO model was trained to detect the position of the basketball hoop within the video frame, which is a key element in determining the scoring condition. The dataset included 3500 images among which are the photos of the basketball court, the surveillance images of the basketball court and the screenshots of basketball matches. This basket detection is crucial because the system must accurately identify the hoop's location to determine if a score has occurred. After the hoop's location is identified by YOLO, the system used frame difference-based motion detection to determine if the basketball has passed through the hoop, thus confirming a scoring event. The paper reports that the YOLO-based hoop detection combined with motion detection runs in real-time and achieves satisfactory scoring detection accuracy (88.64% among 5 test videos). The experiments conducted on real-scenario basketball court videos validated the effectiveness of the proposed method.

2.6 ByteTrack

The advancements in tracking algorithms such as **ByteTrack** complement the capabilities of YOLO by enhancing tracking accuracy and efficiency in complex scenarios like sports. ByteTrack, which utilizes a simple yet highly effective association mechanism based on the intersection over union (IoU) metric, builds upon YOLO's detections to maintain robust track identities even in challenging conditions where occlusions and fast movements are common. This is particularly valuable in sports analytics, where accurately tracking the rapid movements of players and balls is crucial.

In the study detailed in [35], the authors first present ByteTrack as an extension of YOLO object detectors, specifically designed for addressing the limitations of Multi-object tracking (MOT). Compared to traditional tracking algorithms that basically discard low-confidence detections to avoid false positives, ByteTrack leverages these detections to improve tracking continuity and reduce identity switches. The core of ByteTrack's methodology is the association of detection boxes across frames. ByteTrack employs the Kalman filter to predict the state of tracked objects (e.g., position and velocity) and uses the Hungarian algorithm to associate detections with existing tracks based on the Intersection over Union (IoU) metric. This step primarily utilizes high-confidence detections. After associating high-confidence detections, ByteTrack uniquely integrates low-confidence detections into existing tracks. This is done by checking if these low-confidence detections have a sufficient IoU overlap with the predicted states of existing tracks. If a match is found, ByteTrack adds these detections to the respective tracks, which helps in maintaining track continuity especially in scenarios where objects are occluded or only partially visible.

As for the object tracking performance, the authors used widely recognized datasets in the multi-object tracking community, such as MOT17 [18] and MOT20 [9] from the MOTChallenge. These datasets include various video sequences with challenging scenarios, particularly with different levels of crowd density and complex occlusions like pedestrians on crowded streets captured with static and moving cameras. Thus, being tested on more densely crowded MOT20 dataset, the results showed that among all other object tracking methods ByteTrack leads with the highest scores in MOTA (77.8%), IDF1 (75.2%), and HOTA (61.3%), indicating its robustness in maintaining accurate track identities and associations.

The paper [8] can be an appropriate example of how ByteTrack helps in tracking fast-moving objects which is particularly common in sports scenes like football and basketball. The authors presented a

new dataset called SportsMOT, that is largely superior to the MOT17 dataset in terms of the number of frames and bounding boxes, and includes dynamic scenes from 3 sports categories, namely football, volleyball and basketball. Being tested on this proposed dataset, ByteTrack demonstrated the decent results of tracking multiple athletes on the scene with the values of 64.1% for HOTA, 95.9% for MOTA and 71.4% for IDF1.

2.7 Practical applications of basketball analytics

This section will be dedicated to actual commercial products in the field of basketball analytics.

For example, HomeCourt [13] is the interactive basketball mobile application, that captures the video of individuals throwing the ball into basketball hoop and gives them instant guided feedback of their shooting performance. This feedback includes the number of attempted and successful shots, their trajectories and positions on the court, providing users with the detailed map of the court with the marked points of shots positions. Another application feature for basketball enthusiasts that could help in improving ball handling skills is dribbling workout sessions. They work in such a way that the users, while dribbling with the ball, look at the screen of their device, which shows him an image from the front camera, where colored points appear in different positions on the screen, which the user has to trigger with his own hands. Each successful trigger adds score to the final result. At the end, the user receives feedback of his ball handling skills based on this exercise. Moreover, after capturing the statistics for every video from the user's practice throwing sessions, the application is capable of building the detailed visualisations of how users shooting accuracy performance changes over time, providing a complete picture of their progress. Despite the fact that the application does not use any physical sensors and does not require any specific, high-quality camera setups, expanding opportunities for basketball enthusiasts to start using this product, the app's creators inform about some limitations and specifications for the users. For example, it works only for hoops with the net and the color of the ball has to be close to orange. Despite the fact that the creators of the application do not disclose the specific technology by which they process video and get detailed shooting statistics from it, it can be assumed that computer vision technology lies at the core of their method.

The application Swish Hoop [14] uses another approach of solving the problem of successful basketball shots detection by integrating sensor technology directly into the basketball hoop. Swish Hoop's smart hoop system uses a series of sensors around the rim and base of the hoop to capture real-time data on every shot made. This data includes the speed of the ball, its angle of entry, and whether the shot was successful or not. Additionally, the system can detect the specific location from where each shot was taken, enabling it to provide players with precise feedback on their shooting patterns and accuracy from different court positions. One of the standout features of Swish Hoop is its interactive feedback mechanism. Once a session is completed, players receive instant feedback through the Swish Hoop mobile app, which displays detailed statistics and visualizations of their performance. These visualizations include heat maps of shooting accuracy across different zones of the court, allowing players to identify strengths and areas for improvement in their shooting technique. Despite its advanced capabilities, the creators of the product claim that the system primarily relies on a robust Wi-Fi connection to sync data between the hoop sensors and the mobile app, which may limit its use in areas with poor internet connectivity. In addition, the presence of a basketball hoop net is the mandatory condition, since the sensor is supposed to be installed exactly on the net.

2.8 Conclusion

Throughout this literature review, the huge impact of computer vision technologies across various domains has been thoroughly explored. In the realm of sports analytics, particularly basketball, computer vision techniques such as OpenPose, YOLO and ByteTrack has led to significant improvements in tasks like players detection, tracking and motion capture, a ball trajectory analysis and etc., providing previously unreachable analytical depth that enhances coaching strategies, more efficient players training and, subsequently, their general performance.

Innovative commercial applications like HomeCourt and Swish Hoop exemplify how these technologies are being successfully translated into user-friendly products that make high-level sports analytics available to a broader audience. This research work done sets a clear direction for this dissertation project, giving state-of-the-art approaches and solutions for the problem of shots and goals detection in basketball, based on the provided video.

This general problem is decomposed into several subtasks, starting with the accurate detection of such objects on the court as the player, the ball and the basketball hoop. The previously mentioned YOLO approach and its latest released and most relevant model YOLOv8, is perfectly suited for this purpose and will be used as the core technology of the shots and goals detection algorithm. Having the precise data on the objects location and size using the YOLO-generated bounding boxes of these objects, the task will be reduced to coming up with the logic for the algorithm for throws and goals detection using the coordinates of bounding boxes.

Since it is anticipated that we will process and analyze video recordings from individual basketball throwing training sessions, which means there won't be many players and a dynamic game on the video recordings, we exclude the task of handling objects occlusions. Therefore, in the early stages of project development, we will not apply ByteTrack approach. However, as the project evolves and if the need arises to improve tracking accuracy of objects on the court (player, ball and basket), ByteTrack might be integrated to provide better reliability of the algorithm for detecting throws and successful shots.

In addition, since OpenPose is mainly used for human pose recognition, and the players precise posture detection is not supposed to help in the task of scoring detection, its use is not of interest for this project.

3 Exploration of Data and Methods

This section will be divided into different stages of the project, and for each stage all methods will be thoroughly discussed and explained.

3.1 Dataset creation

The initial phase of my project focused on assembling a diverse and robust dataset essential for training an effective YOLO model for basketball analysis. The variety in the dataset is crucial as it impacts the model's ability to accurately detect key elements such as players, basketballs, and hoops.

3.1.1 Data collection

Speaking about dataset content, I sourced video data from two primary channels: YouTube Shorts, which provides a wide variety of basketball gameplay scenarios, and videos recorded by external participants from their basketball training sessions. All included YouTube Shorts videos (14 videos) were recorded with camera installed on some height using the tripod or cameraman, while all videos provided by external contributors (14 videos) were recorded with their phone located on the ground, from different positions on the court. Thus, a total of 28 videos have been uploaded to the Roboflow, where each video was cut with some frequency which depended on the video duration. This frequency was usually adjusted so that the number of frames extracted from the video was in the range from 50 to 400. The total number of extracted frames reached 4,900. Considering the fact that the extracted frames represented basketball scenes from different angles of the court, from different heights of the installed camera, with different degrees of dynamism and different environments (indoors and outdoors), it is expected that the final dataset will be diverse, comprehensive and suitable for building a highly accurate and efficient YOLO model capable of detecting desired objects in different environments. All detailed information about the video recordings that make up the final dataset is presented in the tables 1, 2.

3.1.2 Data labelling

All extracted frames were manually labelled with Roboflow labelling tool, drawing bounding box annotations for the objects on the frame of 3 classes: person, ball and basket. Thus, the total number of annotations for all frames reached 16,103 annotations.

3.1.3 Data splitting

For each video, after it has been cut into the frame and all the objects of classes "person", "ball", "basket" in the frame have been labelled, these frames were shuffled and split into 3 subsets: training set (70%), validation set (20%) and testing set (10%). This way of splitting the dataset was proposed

Video Num	Video Category	Type of Action	Camera Location	Frames Extracted	Ball Color	Environment
1	YouTube Shorts	Dribbling, Layups	Height	205	Orange	Indoors
2	YouTube Shorts	Dunks	Height	66	Orange	Indoors
3	YouTube Shorts	Layups	Height	249	Orange	Indoors
4	YouTube Shorts	Layups	Height	111	Orange	Indoors
5	YouTube Shorts	Dribbling, Throws	Height	110	Orange	Indoors
6	YouTube Shorts	Throws, Layups	Height	298	White red blue	Outdoors
7	YouTube Shorts	Layups	Height	73	Black	Outdoors
8	YouTube Shorts	Dribbling, Throws	Height	171	Orange	Outdoors, Indoors
9	YouTube Shorts	Throws	Height	185	Orange	Outdoors, Indoors
10	YouTube Shorts	Throws	Height	129	Orange	Outdoors
11	YouTube Shorts	Dribbling, Layups	Height	83	Orange	Outdoors
12	YouTube Shorts	Dribbling, Throws, Dunks	Height	204	Orange, Red blue	Outdoors
13	YouTube Shorts	Dribbling, Throws, Dunks	Height	377	Orange	Outdoors
14	YouTube Shorts	Dribbling, Throws	Height	106	Orange	Indoors
15	External Contributor	Dribbling, Throws	Ground	154	Orange	Indoors
16	External Contributor	Throws	Ground	68	Orange	Indoors
17	External Contributor	Dribbling, Throws, Layups	Ground	123	Orange	Indoors
18	External Contributor	Throws	Ground	124	Orange	Indoors
19	External Contributor	Dribbling, Throws, Layups	Ground	232	Orange	Indoors
20	External Contributor	Dribbling, Throws, Layups	Ground	168	Orange	Indoors
21	External Contributor	Dribbling, Throws, Layups	Ground	189	Orange	Indoors
22	External Contributor	Dribbling, Throws, Layups	Ground	239	Orange	Indoors
23	External Contributor	Dribbling, Throws, Layups	Ground	244	Orange	Indoors
24	External Contributor	Dribbling, Throws, Layups	Ground	245	Orange	Indoors
25	External Contributor	Dribbling, Throws, Layups	Ground	236	Orange	Indoors
26	External Contributor	Dribbling, Throws, Layups	Ground	195	Orange	Indoors
27	External Contributor	Throws	Ground	157	Orange	Indoors
28	External Contributor	Throws	Ground	159	Orange	Indoors
Total	-	-	-	4900	-	-

Table 1: Dataset Overview of Basketball Video Sources

Video Num	Link
1	https://www.youtube.com/shorts/PNWtLkEbjUo
2	https://www.youtube.com/shorts/SAVKpoH2YPs
3	https://www.youtube.com/shorts/3UEitmXTNAg
4	https://www.youtube.com/shorts/5PXdig269Q8
5	https://www.youtube.com/shorts/gPkBpZhvDyE
6	https://www.youtube.com/shorts/rSBqoHGfztM
7	https://www.youtube.com/shorts/G1WawtWaGzA
8	https://www.youtube.com/shorts/lLyIqHKH7nw
9	https://www.youtube.com/shorts/FeprGhJlNvw
10	https://www.youtube.com/shorts/bOVbkELNbmI
11	https://www.youtube.com/shorts/w1YRG1UhgPY
12	https://www.youtube.com/shorts/CLPT8LjQ8rc
13	https://www.youtube.com/shorts/l8MyPg5Xk2c
14	https://www.youtube.com/shorts/5sUhFYATYeM

Table 2: Links to Basketball Video Sources

by Roboflow by default. The training set is the largest subset and is crucial for the initial learning process of the model. A larger training set will provide more examples and diverse basketball game scenarios , which will help the YOLO model to learn better and, subsequently, have better detection performance. The validation set is a separate section of our dataset that we will use during training to get a sense of how well our model is doing on images that are not being used in training. By using 20% of the data, there is a substantial amount of information to validate the effectiveness of the model under different conditions and environments. The test set is crucial for assessing the final YOLO model's performance, and it was decided to leave 10% of the data for it. It acts as new, unseen data to simulate how the model will perform in real-world conditions.

The chart 3 represents a stacked bar chart that visualizes the distribution of annotated objects by class (ball, basket, person) across different splits of a dataset. Each bar represents a class of objects, and the segments within each bar indicate the number of instances of that class in each dataset split. The total height of each bar represents the total number of instances of that class across all dataset splits. As you can see from the graph, the 'person' class has the highest number of total instances (6193),

Figure 3: Counts of Instances by Class and Subset

the 'basket' class has the second-highest count (5530), while the 'ball' class has the fewest number

of instances (4380) across all the images. Such a picture of the distribution of the total number of instances can be observed, since often a lot of people appear in video fragments taken from YouTube Shorts, at the same time the number of balls or basketball hoops rarely exceeds 2. Confirmation of this can be clearly seen in graph 4, describing in more detail how many objects of each class are present in the images. Approximately the same large number of instances per each class reduces the risk that the model will be biased, meaning it will not show less accurate detection in favor of a certain object class.

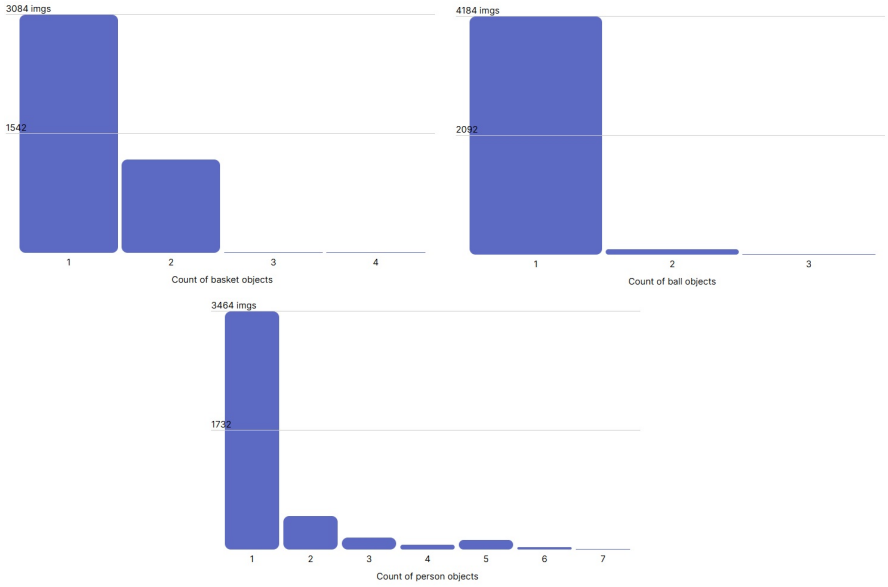

Figure 4: Counts of Objects by class in all the frames

3.1.4 Data Preprocessing and Augmentation

Image preprocessing is the steps taken to format images before they are used by model training and inference. It can include image resizing, rotation, color corrections, etc. In our case we use Auto-Orient and resize transformation, stretching all images to 1280x1280 pixels. Auto-Orient ensures that all images are aligned in the same orientation, which is crucial for the YOLO model to learn correctly. Stretching the frames will provide the correct model training, since YOLO model architecture requires the input images to be the same size, while the resolution of collected frames may vary. Moreover, in the next stage we will train our YOLO model on the same image resolution of 1280x1280, since such high image resolution may benefit the resulting model's detection capabilities of such small objects on the frame as a ball, and a basket.

Image augmentation involves modifying images to generate varied versions of the same content, thereby providing the model with a broader range of training examples. For instance, by randomly changing an image's rotation, brightness, or scale, the model is trained to recognize the subject of the image under diverse conditions. While image preprocessing steps are applied to training and test sets, image augmentation is only applied to the training data. Thus, I applied the horizontal flip manipulation to help the model be insensitive to subject orientation. At the same time, it increased the training set size by 1.75 times, from 3431 to 6017 images, which is good for training more accurate model.

After applying all mentioned image manipulation techniques to the dataset, I prepared the final version of the dataset with the distribution of the number of images across subsets as follows:

- **Training set (80%)**: 6017 images
- **Validation set (13%)**: 981 images
- **Testing set (7%)**: 488 images

3.2 Model Training and Evaluation

3.2.1 Model Training

This section will describe the process of training and evaluating the YOLO model based on the dataset we have prepared.

Since I develop a web application that will eventually be deployed on a cloud server, and all computing processes will also be performed on a cloud server using cloud computational resources like GPU, it was decided to use the latest released, highly efficient, yet fast and small in number of parameters YOLOv8l (large version), as a pre-trained model, that will be used as the starting point for training. The YOLOv8 models, provided by Ultralytics, were trained from scratch on famous COCO dataset, containing 200,000 labeled images with over 1.5 million object instances across 80 categories. Among all the classes, there are classes "Person" and "Ball", which intersects with the classes in our custom YOLO model, that will certainly benefit the resulting performance of our model and reduce training time requiring less computational resources. Moreover, training deep learning models from scratch generally requires large amounts of diverse data to perform well without overfitting, and since our dataset is small relative to the COCO dataset, it was decided to use the training approach using a pre-trained model.

In the training of the YOLO model, two critical settings were employed to optimize performance. The model was trained for 100 epochs, with each epoch representing a complete pass through the entire training set, allowing the model to iteratively learn and adjust from the data over a total of 100 cycles. Additionally, the image size was set to 1280 pixels for both height and width during training. This uniform image size is crucial for maintaining consistency throughout the training process. Particularly, the choice of 1280 pixels is strategic for enhancing the model's capability to detect smaller objects, such as the "ball" in our dataset, which is relatively small compared to "person" class. This larger resolution helps in capturing finer details, crucial for accurate detection of small objects.

As for the computational resources, I utilized the powerful A100-SXM4-40GB GPU with 40514 MiB of memory, a resource provided by Google Colab Pro. This high-performance computing environment significantly enhanced and speed up the training process. During the entire training process, the GPU memory remained stable around 40GB, reflecting efficient utilization without overload.

Throughout the training, each of 100 epochs consistently evaluated the model performance on validation set (981 images and 3219 instances). The graph 5 illustrates the training and validation performance metrics of our custom YOLOv8l model over 100 epochs. Across all the 100 epochs, key

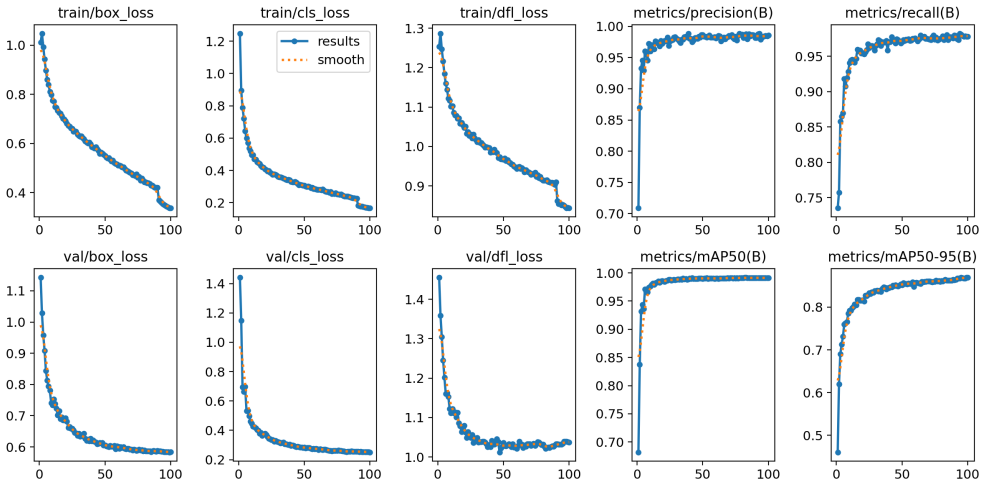

Figure 5: Training results

performance metrics, such as box loss, class loss, and distribution focal loss, all showed a downward trend, indicating enhanced accuracy in predicting both the locations and classifications of objects within images. The precision and recall graphs remain remarkably stable and high across the training epochs. High precision indicates that the model makes accurate predictions with a low rate of false positives, while the high recall indicates that the model successfully identifies a large proportion of

relevant instances. The mean Average Precision (mAP) metrics, both mAP50 and the more stringent mAP50-95, demonstrate increasing accuracy, highlighting the model's growing ability to detect objects accurately across various Intersection over Union (IoU) thresholds. Overall, all mentioned evaluation metrics indicated a successful training progression, with the model becoming increasingly precise and reliable in its object detection capabilities. The entire training process took 5,902 hours of continuous training.

3.2.2 Model Evaluation

After completing the training, I evaluated the effectiveness of object detection by our model on a test dataset (488 images, 1612 instances) that the model had never seen before.

The graph 6 represents confusion matrices showing the results of evaluating a YOLO model's performance on a testing set, presented in both absolute counts and normalized forms, for object classifications into categories such as 'ball', 'basket', 'person', and 'background'. In the absolute counts matrix (left matrix), the model demonstrates high accuracy with predominant correct classifications for 'ball' (419 correct), 'basket' (546 correct), and 'person' (628 correct), with minimal misclassifications between these categories but some instances misclassified as 'background'. The normalized matrix further emphasizes this precision, highlighting nearly perfect identification of 'person' (1.00), excellent accuracy for 'basket' (0.99), and 'ball' (0.97). However, it also reveals minor issues with false positives, where some objects are mistakenly identified as 'background'. This might be due to the fact that the dataset did not include background images, which do not contain objects and which are commonly used to minimize false positives. Overall, based on confusion matrices, the model exhibits strong object classification performance across classes "person", "basket" and "ball" with minimal error.

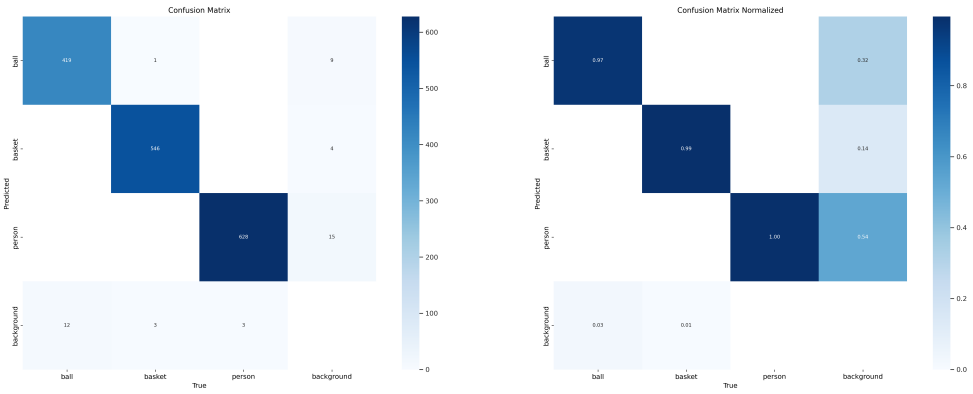

Figure 6: Object Detection Performance: Confusion Matrices Overview

The graphs in 7 represent various performance metrics in the form of curves for our YOLOv8l model evaluated on a testing set, specifically illustrating F1 score (a harmonic mean of precision and recall), precision, and recall against confidence levels for detecting objects categorized as 'ball', 'basket', and 'person'. The F1-Confidence curve indicates that the model achieves near-perfect F1 scores (0.99) at a confidence threshold of about 0.6, reflecting a strong balance between precision and recall. The Precision-Confidence curve shows high precision even from low confidence levels, suggesting minimal false positives, while the Recall-Confidence curve maintains high recall over almost the entire range of confidence level which indicates that the model consistently identifies most of the relevant objects across all categories with a high degree of certainty, regardless of the confidence setting, until it becomes excessively stringent (after 0.9 and up to 1). The Precision-Recall curve demonstrates that the model retains high precision and recall across the spectrum for all categories, underscoring its robust detection capabilities.

The table 3 displays the numerical results from evaluating our custom YOLOv8 model on a testing set, providing metrics for overall performance as well as performance by individual classes: ball, basket, and person. Precision scores for the bounding boxes are consistently high, almost reaching 1.0, indicating that the model precisely identifies the bounding boxes containing the targeted objects. Recall metrics are similarly high, showcasing the model's ability to capture nearly all relevant objects. The mean Average Precision (mAP) at an IoU threshold of 0.50 is nearly perfect for all classes,

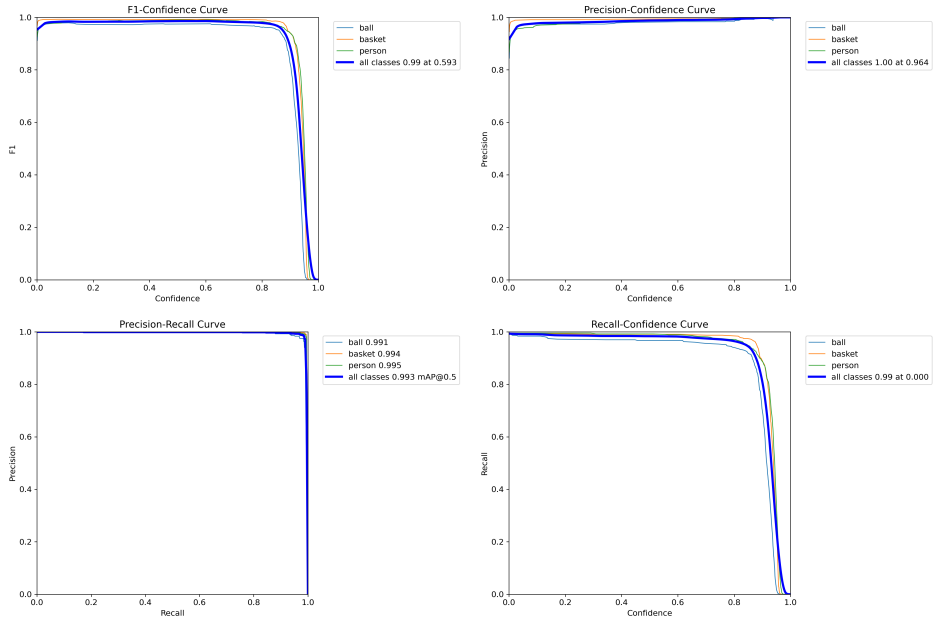

Figure 7: Object detection performance curves for YOLO model: F1, Precision, Recall

indicating excellent model accuracy at this threshold. When the criteria are tightened across IoU thresholds from 0.50 to 0.95, the mAP scores are slightly lower but remain highly robust, confirming the model's reliable detection capabilities even under more stringent conditions.

Class	Images	Instances	Precision	Recall	mAP50	mAP50-95
all	488	1612	0.989	0.982	0.993	0.878
ball	488	431	0.983	0.968	0.991	0.807
basket	488	550	0.994	0.991	0.994	0.934
person	488	631	0.989	0.989	0.995	0.893

Table 3: Performance metrics of YOLO model on testing dataset across different classes

3.2.3 Model Inference

Picture 8 demonstrates objects detection capabilities on completely unseen data. As can be observed, in this case it perfectly detects actual objects on the provided frames: person, basket and ball.

3.3 Throws and goals detection algorithm

3.3.1 Building rectangular zones

In general terms, the logic of the algorithm for detecting throws and successful shots is based on obtaining and managing the coordinates of bounding boxes of the player, the ball, and the basket from each frame, as the output of our custom YOLOv8 model, and detecting the "ball" object in 3 certain rectangular zones built next to the basket. Before explaining the logic of the algorithm in detail, it is worth explaining what these zones are and how they are constructed based on the position of the basketball hoop in the frame.

For building these zones, I used PolygonZone from "supervision" Python library, a class for defining a polygon-shaped zone within a frame for detecting objects. Each of these 3 rectangular zones performs its own function and is constructed as follows:

- For successful shot detection:

  1. Zone above the basket (zone_above)

Figure 8: Model inference on unseen data

- Zone width: the width of the "basket" bounding box increased on 10% (5% from both left and right sides).
- Zone height: the height of the "basket" bounding box.
- Zone location: right on top of the "basket" bounding box.

2. Zone below the basket (zone_below)
   - Zone width: the width of the "basket" bounding box decreased on 40% (20% from both left and right sides).
   - Zone height: the height of the "basket" bounding box.
   - Zone location: shifted down by half the height of the "basket" bounding box.

- For throw detection:

1. Full basket coverage zone (zone_general)
   - Zone width: the width of the zone_above increased on 100% (50% from both left and right sides).
   - Zone height: the top of the zone_above is the upper boundary of the zone_general, and the bottom of the zone_below is the bottom of the zone_general.
   - Zone location: based on the locations of zone_above and zone_below, so it covers all zones including the "basket" bounding box

Figure 9 showcases how these zones look like at the frames with all other detected objects presented. As can be seen, 3 zones are built for each basket, regardless of the number of baskets in the frame.

3.3.2 The logic of the algorithm

The algorithm leverages the YOLO model to analyze each frame from the provided video for detection of such objects as a "person", "basket" and "ball", and then analyze the interaction of these objects and predefined rectangular zones located next to the basketball hoop.

The schema 10 represents how each frame is being processed by our algorithm. Initially, if it is the first frame, the function sets up all 3 zones for each "basket" object, and then for each type of zone (zone_above, zone_below, zone_general) it selects the one with the largest area. Obviously, we start with the values of the number of shots and goals equal to 0. For every frame, the algorithm runs YOLO model and collects detected objects. Then, we check if the "ball" object intersect with any

Figure 9: Examples of building zone_above, zone_below, zone_general

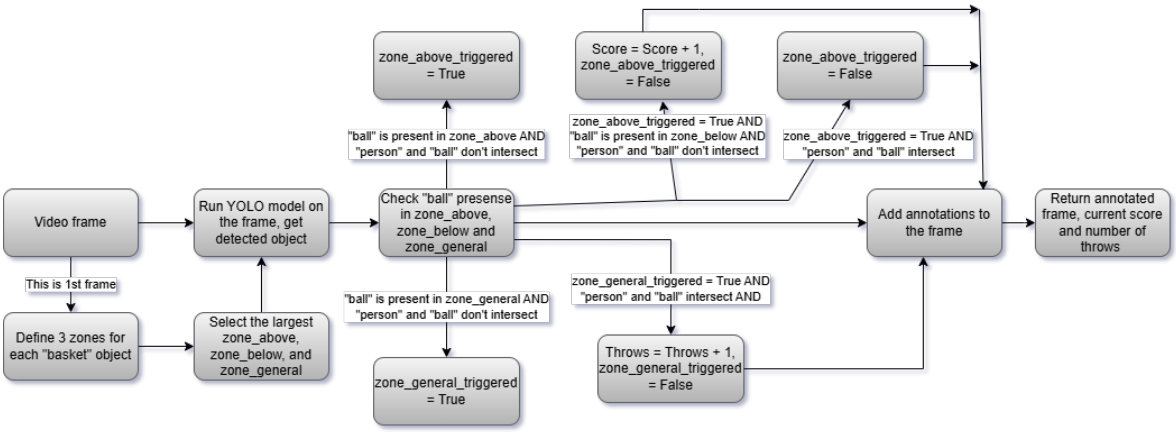

Figure 10: Pipeline diagram of how every frame is being processed by the algorithm

predefined zones without intersecting with the person. I use special variables "zone_above_triggered" and "zone_general_triggered" to truck whether zone_above and zone_general were triggered by the "ball". As soon as the ball hits these zones, these variables become true and will keep these values for all other frames until we set them as false.

Further, speaking about the logic of a scoring detection, if the variable "zone_above_triggered" is true, we are waiting for the moment when the ball appears in the zone_below. If this happens before the moment when the bounding boxes of the person and the ball intersect, we add the "score", meaning the number of successful shots, by one. Otherwise, if the outcome of intersection between a person and a ball occurs earlier, we do not change value of "score". As a result of both of these outcomes, we change the value of the variable "zone_above_triggered" to false, signifying the end of a circle of the throw.

The logic of a throw detection is similar; if the variable "zone_general_triggered" is true, we are waiting for the moment when bounding boxes of the person and the ball intersect. As soon as this happens, we add the value for throws by one, indicating the end of the circle for this throw. And this outcome is unambiguous, since a person needs to pick up the ball in order to make the next throw, then their bounding boxes will intersect.

After all the frames from the video have been processed, and the totals for the number of shots

and goals were obtained, we can easily calculate the number of misses:

$$\text{misses} = \text{throws} - \text{goals}$$

3.3.3 Video Requirements

Based on the capabilities of the YOLO trained model and the logic of the algorithm, to obtain the most accurate throws and goals detection results, it is anticipated that the provided video will match following requirements:

1. The video must end with a frame of a person holding a ball.
2. During the throw, a person should not stand in front of the camera and obscure the visibility of the basketball hoop. This may lead to inaccurate results.
3. If there are several basketball hoops in the video, shots and goals will be counted in the largest hoop in the frame.
4. There must be a net on the basketball hoops throughout the video.
5. There must be only 1 ball throughout the video.
6. The color of the ball must be orange or at least close to it.
7. The environment in the video can be both indoors and outdoors.
8. The camera must be stationary throughout the entire video recording.
9. The camera can be positioned on the ground or mounted at some height using a tripod.
10. The camera must be in positions on the court marked with red dots (Picture 11).

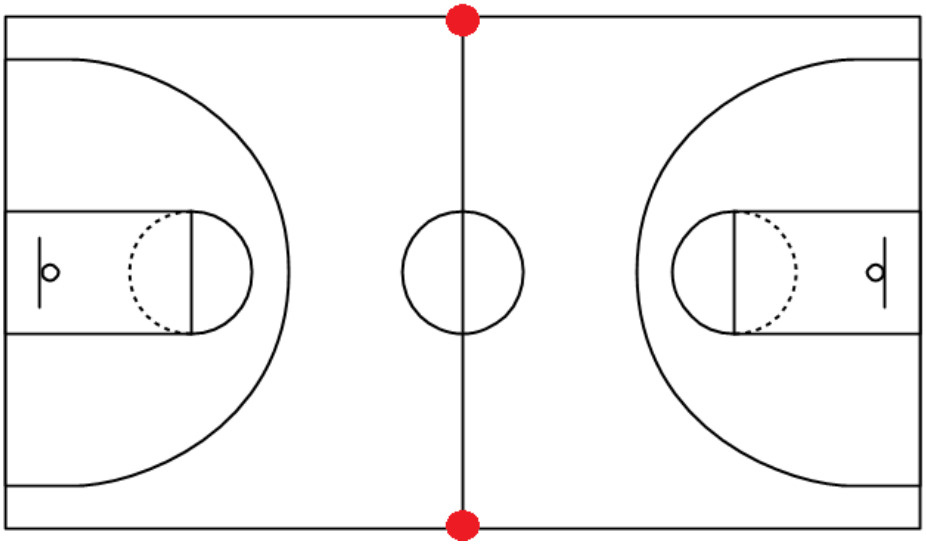

Figure 11: Required camera positions on the court marked with red dots

4 Proposed Future Analysis

This section will be devoted to further work on the project and possible improvements for it will be considered. Here we explore how integrating advanced tracking technology like ByteTrack can improve our system's ability to detect throws and goals accurately during basketball training sessions. We also plan to test the system thoroughly using videos taken from different camera angles and locations,

both indoors and outdoors. These tests will help us make sure our system works well under various conditions and help identify any issues when the environment changes. The results will guide us in refining the system before integrating it into a web application that athletes can use to analyze their performance.

4.1 Applying ByteTrack

Integrating ByteTrack into the current algorithm could significantly enhance the accuracy of detecting throws and goals by maintaining stable tracking of the ball and the player across video frames. ByteTrack is able to manage object identities consistently, even through occlusions or rapid movements, which is quite common for video recordings from basketball training sessions. It helps in minimizing detection errors such as false positives and negatives by ensuring the ball's trajectory and player interactions are continuously monitored. This might result in more reliable scoring and goal detection as ByteTrack ensures that the ball's presence in specific zones is accurately tracked over time.

4.2 Evaluation of throwing and scoring algorithm

This section will discuss one of the ways to objectively and accurately assess the effectiveness of the algorithm. Initially, I plan to record and prepare 1-2 minute videos myself, which the model has not seen before during its training and validation processes, following these instructions:

1. videos recorded indoors (8 videos in total):
    (a) camera is in the first possible position (the red dot on the lower boundary of the court in Figure 11)
        i. camera is located on the ground (2 videos):
            A. video includes short distance throws
            B. video includes long distance throws
        ii. camera is mounted at some height using a tripod (2 videos):
            A. video includes short distance throws
            B. video includes long distance throws
    (b) camera is in the second possible position (the red dot on the upper boundary of the court in Figure 11)
        i. camera is located on the ground (2 videos):
            A. video includes short distance throws
            B. video includes long distance throws
        ii. camera is mounted at some height using a tripod (2 videos):
            A. video includes short distance throws
            B. video includes long distance throws
2. videos recorded outdoors (8 videos in total):
    (a) camera is in the first possible position (the red dot on the lower boundary of the court in Figure 11)
        i. camera is located on the ground (2 videos):
            A. video includes short distance throws
            B. video includes long distance throws
        ii. camera is mounted at some height using a tripod (2 videos):
            A. video includes short distance throws
            B. video includes long distance throws
    (b) camera is in the second possible position (the red dot on the upper boundary of the court in Figure 11)
        i. camera is located on the ground (2 videos):
            A. video includes short distance throws

B. video includes long distance throws

ii. camera is mounted at some height using a tripod (2 videos):

A. video includes short distance throws

B. video includes long distance throws

Thus, total number of videos will be 16. Each of this 1-2 minute video will include 10-20 shots. After collecting all these videos, I will run our algorithm for each of them, obtaining number of throws, goals and misses, as a result. Then we will manually calculate actual values of throws, goals and misses for each video and compare with the results of the algorithm, thereby obtaining the accuracy and effectiveness of the proposed framework.

By employing videos that cover various camera positions and setups, both indoors and outdoors, the testing framework ensures a comprehensive evaluation across a broad range of playing conditions. This methodology not only verifies the algorithm's robustness and accuracy in real-world environments but also helps in identifying specific conditions under which the performance might degrade. This will be very useful, as this framework will then be integrated into the web application and tested by unbiased users.

4.3 Web application development

The final phase of the project centered on the development and deployment of a cloud-based web application. This application will provide a user-friendly interface where athletes can upload their training videos and receive the clear statistics of their shooting performance. Users will benefit from interactive visualizations, such as line graphs showing changes in shooting accuracy over time and bar charts showing the distribution between successful and unsuccessful throws. Additionally, we will assess how the application impacts users' training habits and shooting performance improvements. The backend, that will be built with the FastAPI Python web-framework, will manage video processing and store personal and basketball statistics in a PostgreSQL database. SQLAlchemy will facilitate interaction between Python and the database through an optional high-level Object-Relational Mapping (ORM) interface, allowing the use of Python classes instead of raw SQL. This architecture supports the intensive computational demands of video processing and ensures secure, efficient management of user data.

4.4 Future work plan

In table 4, you can see a detailed plan for further work, distributed by weeks. A green cell for a task means that work will be done for that specific task this week.

ID	Task	June-24		July-24				August-24			
		W 3	W 4	W 1	W 2	W 3	W 4	W 1	W 2	W 3	W 4
1	Applying ByteTrack										
2	Evaluation of Throwing and Scoring algorithm										
3	Web Application Development										

Table 4: Future work plan

5 GitHub

Here is the link to GitHub repository of this project: GitHub

References

[1] David Acuna. "Towards real-time detection and tracking of basketball players using deep neural networks". In: *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA*. 2017, pp. 4–9.

[2] Ilhan Aydin and Nashwan Adnan Othman. "A new IoT combined face detection of people by using computer vision for security application". In: *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*. IEEE. 2017, pp. 1–6.

[3] Alex Bewley et al. "Simple online and realtime tracking". In: *2016 IEEE international conference on image processing (ICIP)*. IEEE. 2016, pp. 3464–3468.

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection". In: *arXiv preprint arXiv:2004.10934* (2020).

[5] Zhe Cao et al. "Realtime multi-person 2d pose estimation using part affinity fields". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7291–7299.

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

[7] Ivan Ćirić et al. "INTELLIGENT COMPUTER VISION SYSTEM FOR SCORE DETECTION IN BASKETBALL". In: *Facta Universitatis, Series: Automatic Control and Robotics* 22.2 (2024), pp. 075–085.

[8] Yutao Cui et al. "SportsMOT: A large multi-object tracking dataset in multiple sports scenes". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 9921–9931.

[9] Patrick Dendorfer et al. "Mot20: A benchmark for multi object tracking in crowded scenes". In: *arXiv preprint arXiv:2003.09003* (2020).

[10] Bauyrzhan Doskarayev et al. "Development of Computer Vision-enabled Augmented Reality Games to Increase Motivation for Sports". In: *International Journal of Advanced Computer Science and Applications* 14.4 (2023).

[11] Xu-Bo Fu, Shao-Long Yue, and De-Yun Pan. "Camera-based basketball scoring detection using convolutional neural network". In: *International Journal of Automation and Computing* 18.2 (2021), pp. 266–276.

[12] Xubo Fu et al. "Multiple player tracking in basketball court videos". In: *Journal of Real-Time Image Processing* 17 (2020), pp. 1811–1828.

[13] HomeCourt. *HomeCourt: The Basketball App*. Available at: https://www.homecourt.ai/. Accessed: 4 June 2024. 2022.

[14] Swish Hoop. *About SwishHoop*. Available at: https://www.swishhoop.com/about. Accessed: 4 June 2024. n.d.

[15] Shihao Hou et al. "A Basketball Training Posture Monitoring Algorithm Based on Machine Learning and Artificial Intelligence". In: *Mobile Information Systems* 2022 (2022).

[16] Chuyi Li et al. "YOLOv6: A single-stage object detection framework for industrial applications". In: *arXiv preprint arXiv:2209.02976* (2022).

[17] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer. 2014, pp. 740–755.

[18] Anton Milan et al. "MOT16: A benchmark for multi-object tracking". In: *arXiv preprint arXiv:1603.00831* (2016).

[19] Banoth Thulasya Naik, Mohammad Farukh Hashmi, and Neeraj Dhanraj Bokde. "A comprehensive review of computer vision in sports: Open issues, future trends and research directions". In: *Applied Sciences* 12.9 (2022), p. 4429.

[20] Sen Qiao, Yilin Wang, and Jian Li. "Real-time human gesture grading based on OpenPose". In: *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. IEEE. 2017, pp. 1–6.

[21] Joseph Redmon and Ali Farhadi. "YOLO9000: better, faster, stronger". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.

[22] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[23] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[24] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems* 28 (2015).

[25] Shuji Tanikawa and Norio Tagawa. "Player Tracking using Multi-viewpoint Images in Basketball Analysis." In: *VISIGRAPP (5: VISAPP)*. 2020, pp. 813–820.

[26] Graham Thomas et al. "Computer vision for sports: Current applications and research topics". In: *Computer Vision and Image Understanding* 159 (2017), pp. 3–18.

[27] Ultralytics. *YOLOv5*. https://github.com/ultralytics/yolov5. Accessed: 12 June 2024. 2024.

[28] Ultralytics. *YOLOv8*. https://github.com/ultralytics/ultralytics. Accessed: 12 June 2024. 2024.

[29] Athanasios Voulodimos et al. "Deep learning for computer vision: A brief review". In: *Computational intelligence and neuroscience* 2018 (2018).

[30] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 7464–7475.

[31] Jianbo Wang et al. "Ai coach: Deep human pose estimation and analysis for personalized athletic training assistance". In: *Proceedings of the 27th ACM international conference on multimedia*. 2019, pp. 374–382.

[32] Ben G Weinstein. "A computer vision for animal ecology". In: *Journal of Animal Ecology* 87.3 (2018), pp. 533–545.

[33] Jiongen Xiao, Wenchun Tian, and Liping Ding. "Basketball action recognition method of deep neural network based on dynamic residual attention mechanism". In: *Information* 14.1 (2022), p. 13.

[34] Young Yoon et al. "Analyzing basketball movements and pass relationships using realtime object tracking techniques based on deep learning". In: *IEEE Access* 7 (2019), pp. 56564–56576.

[35] Yifu Zhang et al. "Bytetrack: Multi-object tracking by associating every detection box". In: *European conference on computer vision*. Springer. 2022, pp. 1–21.