Real-Time Dynamic Trajectory Analysis of Spherical Projectiles: a YOLO-Based Basketball Tracking System

Eshan Shukla*, Kushagra Sinha*, Jash Khatri*, and Supriya Agarwal
*Dept. of Computer Engineering, SVKM's Narsee Monjee Institute of Management Studies - Mukesh Patel School of Technology Management & Engineering, Mumbai, India
Email: eshanshukla2507@gmail.com, kushneedsanemailid@gmail.com, khatri.jash2005@gmail.com, Supriya.Agrawal@nmims.edu

***Abstract*—The given paper provides a variant of a comparative analysis of three lightweight and real-time object detectors, namely, YOLOv8n, YOLOv5n, and a MobileNet-based SSD, when it comes to a specific and challenging object-finding task a basketball hoop localization. The models were tested on a custom, highly imbalanced data set where the two classes of the data have a lot of inter-class similarity (2.2:1 instances:ratio). Training all models was done on an NVIDIA RTX 3050 GPU. YOLOv8n with a training of 25 epochs slightly surpassed the other models with the highest mAPs at 0.5. YOLOv5n with 25 epochs training had a marginally less mAP at 0.5 of 0.771. Both versions of YOLO were badly affected by ambiguity in classes; the v8n model confused 50 percent of instances of the basket class to be rim, and the v5n model confused 56 percent. The version of the MobileNet-based model, which was also trained on 25 epochs, was fundamentally inappropriate to this fine-Grained task, and gives a non-viable mAP@0.5 of 0.672 and results in disastrous class confusion (67 percent of the baskets misclassified). This paper finds that despite the promising training efficiency of modern nano-architectures such as YOLOv8n and a modest accuracy advantage, ambiguous and imbalanced nature of the data is the breadth of the performance, and even the SOTA lightweight models can only be performed marginally without providing serious data-centric mitigation policies.**

***Index Terms*—Object Detection, YOLOv8, YOLOv5, MobileNet, Sports Analytics, Computer Vision, Lightweight Models, Dataset Imbalance**

I. Introduction

The confluence of high-performance computing and advanced machine learning has catalyzed a data-driven renaissance across numerous industries. Among these, the field of sports analytics has undergone one of the most profound transformations, evolving from simple statistical tallies to a complex ecosystem of real-time vision-based data extraction [1]. The modern sports world engages in a "data arms race" of sorts, where the ability to acquire and understand the information gathered on the cameras gives the users a direct advantage. This paradigm shift is a direct result of the recent growth of computer vision, which enables various activities from automated event tagging to tactical heat mapping to real-time biomechanical analysis allowing players to avoid injury. In this high-stakes field, basketball sports have a special case of an analytical problem. Although there are now fairly mature systems that provide the means of player and ball tracking, the automatic analysis of small, critical-event main focus structures critically important to the sport, namely the hoop assembly, is yet to be an easy endeavor. The hoop is the most important in the court of non-players because it determines all the scoring. Nevertheless, it is not a unitary body. It is a combination of two different items, the hard, metallic rim, and the soft, net like basket. Advanced analytics, like verifying shot passages, simulating the physics of rim-interactions (e.g. roll-out), or verifying swish basketball shots that have no physical interaction with the net, should divide between these two components. This high-resolution detection task is foolingly complicated. To begin with, there is extreme similarity between the objects of different classes; there is co-location between the two different objects: the definition of a basket is that it is attached to the rim, and it occupies a similar spatial location. Second, there is a large intra-class variance in the basket class, with the appearance of the deformable net varying radically depending on the interaction, the momentum and the camera angle. Third, the hoop assembly is frequently and high speed occluded by ballcarrying and attempting ballplayers and also highly mobile by motion blur, by broadcast camera scurrying. These difficulties make classical computer vision methods based on the use of fixed, hand-designed feature representations such as HOG [2] or SIFT [1] completely useless. The appearance of Deep Convolutional Neural Networks (DCNNs) created a way out. Nevertheless, the eminent architectures have traditionally been in two categories which is not a preferred trade-off. Two-stage detectors including the R-CNN family [4]–[6] are very accurate as regions are proposed and later classified. They have high computational complexity, but it comes at a very high cost; multi-stage pipeline and long indicative latency makes them essentially unsuitable to the real time requirements of live sports analytics of 30 or more frames per second (FPS). This weakness has led to the creation of single stage detectors, including SSD [11] and the You Only Look Once (YOLO) family [7], which view detection as a single, unified regression problem. These architectures provided a breakthrough in inference speed, finally making real-time detection viable. This paper focuses

on the modern, lightweight "nano" variants of these models (e.g., YOLOv8n, YOLOv5n [8]) and mobile-centric backbones (e.g., MobileNet [10]), as these represent the most practical solution for scalable deployment on consumer-grade (e.g., an NVIDIA RTX 3050) or embedded edge-computing hardware. This study, however, is not a simple performance benchmark. The primary contribution of this work is a critical comparative analysis of these SOTA lightweight models under pragmatic, non-ideal conditions. We intentionally stress-test YOLOv8n, YOLOv5n, and a MobileNet-SSD on a custom-compiled, real-world dataset characterized by two severe, endemic flaws: a 2.2:1 class imbalance and the high semantic ambiguity detailed above. Our goal is to move beyond sterile benchmark scores (e.g., on the COCO dataset) and rigorously dissect the failure modes of these architectures. We seek to answer: How do differences in architecture, such as the anchor-free design of YOLOv8, impact performance on such a task? Is training efficiency really a factor? If yes, then how exactly does differing training efficiency matter (YOLOv8 having 25 epochs only, while YOLOv5 went through 25 epochs). And finally the most important: are these SOTA models actually the solution or is the flawed dataset an unconquerable issue? The upcoming Sections review the overall evolution of object detection models and their application in the world of sports. Then the next section after that goes through a deep-dive into a detailed examination of our dataset's structure and the exact setup used by each of our models. Section IV follows up with an analysis of our comparative work, focusing primarily on the metrics and the insights derived from the confusion matrices. Finally, Section V concludes the paper with a summary of the findings and a proposal for a data-centered shift for future work in this domain.

II. Related Work

The heart of this paper lies in the convergence of three separate domains, namely: the ongoing evolution of object-detection architectures, the push toward real-time lightweight models, and the growing use of computer vision in sports analytics. A further review of these areas is critical to position our overall contribution.

A. The Paradigm Shift: From Hand-Crafted to Learned Features

The first object detection techniques were defined by approaches that use hand-crafted feature descriptors, including Scale-Invariant Feature Transform (SIFT) [1] and Histograms of Oriented Gradients (HOG) [2]. These were later combined with a sliding-window comparator such as a Support Vector Machine (SVM), but were prone to utter brittleness. They were unable to generalize to lighting, scale, pose, and occlusion variations, which are inherent in the live sports broadcasting environment which is not controllable. With AlexNet [3], these conventional approaches became outdated so to speak because of the so-called "deep learning revolution." The first paradigm shift was the introduction of the R-CNN (Regions with CNN features) [4]. R-CNN used a two-step approach: firstly, it was used to create region proposals through selective search and then it classified these regions with the help of the Convolutional Neural Network (CNN). Although correct, its inferencing time (in seconds per frame) was too crippling to be considered in a real-time application. The iterative models such as Fast R-CNN [5] and Faster R-CNN [6] radically accelerated execution through computation sharing and a dynamically forecasted Region Proposal Network (RPN) respectively. Although all these advances were made, the underlying two-stage pipeline was still a computational bottleneck.

B. The Real-Time Revolution: Single-Stage Detectors

A second revolution was the creation of single-stage detectors, that is, those that do not require the model to perform any region proposal step but instead directly regress to predict bounding boxes and class likelihoods on the entire image in a single pass.

1) The YOLO Family: The most important single-stage architecture has been the so-called YOLO (You Only Look Once) family proposed by Redmon et al. [7]. The initial YOLOv1 was groundbreaking with the speed, yet had a problem with objects that were small and localization. Later versions (YOLOv2, YOLOv3) have added anchor boxes and fully connected feature pyramid networks to enhance performance. Our study focuses on two of its most modern, lightweight descendants:

- **YOLOv5:** Developed by Ultralytics [8], YOLOv5 was one of the most commonly used architectures because of its remarkable usability, embedded augmentation pipelines, and unambiguously scaled (n/s/m/l/x models). The `YOLOv5n` (nano) model, especially, established a new benchmark regarding performance on low-power edge devices and CPU. It is a very optimized anchor-based baseline.
- **YOLOv8:** A more recent release by Ultralytics [9] that is a major architectural change. It abandons the legacy of its predecessors, which is anchor-based, and merits an anchor-free detection head, a novel C2f (CSPDarknet-to-feature) backbone and additional optimizations. The theory of this change is to enhance robustness against objects with different aspect ratios. Its direct competitor is the `YOLOv8n`, which will have a higher accuracy and efficiency compared to `YOLOv5n`.

A close comparison between the `v5n` and the `v8n` using the same compromised dataset in this paper offers a new task-oriented assessment of these two significant differences in architectural features.

2) Mobile-Centric Architectures (MobileNet-SSD): Concurrent with the YOLO family, which is an independent line of work concerned with the development of hyper-efficient backbones on mobile and embedded devices. The most notable of them is the MobileNet family [10], which added depth-wise separable convolutions in a bid to significantly reduce the number of parameters and computational cost (FLOPs). This efficient backbone is most commonly paired with the Single Shot MultiBox Detector (SSD) [11] architecture. An

`MobileNet-SSD` represents a different philosophical approach: it prioritizes minimal computational footprint above all else. It serves as our third contender to answer: at what point does architectural efficiency result in a catastrophic loss of representational power, rendering a model incapable of solving a fine-grained, ambiguous task?

C. The Application Gap: Vision in Sports

The application of computer vision in sports is a well-established field [13]. A vast body of work exists on the "solved" problems of player detection and ball tracking, often using older methods like Kalman filters or, more recently, dedicated tracking-by-detection models [14]. However, the literature is significantly thinner on the *fine-grained classification of co-located components*. Most work treats "the hoop" as a single entity, if it is detected at all. This paper addresses this specific gap: we are not asking "can a model find the hoop?" but rather, "can a modern, lightweight model *differentiate* the components of the hoop assembly under non-ideal conditions?" This performance loss that we have documented across all three models shows that this is a surprisingly difficult challenge to overcome that the field itself has not properly addressed, showing how the model design and data quality interweave together to drive success.

III. Methodology

The experimental nature of the proposed study is designed in such a way that, it can give a direct and formidable comparison of the three lightweight architectures that are to be studied in this study. This section describes how our novel data is configured, the exact hardware and software environment, the training specifics of each of our models, and evaluation metrics.

A. Dataset Acquisition and Analysis

The models were trained on a custom dataset, which was compiled manually as well and obtained from high-definition broadcast footage of professional basketball games. The data has been structured to represent a large range of in-play situations such as a variety of camera angles, light (e.g. indoor arenas, outdoor courts), and amount of player and ball occlusion.

1) Labeling and Composition: The images were annotated by hand using CVAT in the YOLO `.txt` format. This fine-grained activity was divided into two classes:

- **rim:** The rigid and circular metallic hoop.
- **basket:** The deformable mesh net attached at the rim.

Initial examination of the finished dataset turned out that there are 2,500 images and split it down into an 80 percent training set and 20 percent validation set, as it is customary to do.

2) Inherent Dataset Biases: The second important aspect of our method was to examine the intrinsic data set biases that we expected to be our major performance constraint. This analysis, as shown in Figure 1, validated two major problems:

1) **Class Imbalance:** The data set is extremely imbalanced. A tally of all labelled instances (Figure 1-a) indicates a number of approximately **1,050** instances of 'rim' as opposed to **480** instances of 'basket'. Such a 2.2:1 imbalance is excessively in favour of the 'rim' class, which poses a major risk that a model will have a predictive bias, which means that it will default to the majority class in cases of uncertainty.
2) **Spatial Co-location:** The correlogram in Figure 1-b indicates that the spatial correlation of the classes of 'basket' and 'rim' is close to being 1. The 'y' (vertical) coordinate is nearly the same and the 'x' (horizontal) coordinate is compactly clustered. This establishes that they co-locate physically and overlap visually providing a "hard" problem of detection to which the model is forced to detect minute **textural** and morphological variations instead of using spatial heuristics.

B. Experimental Environment and Training

To ensure a fair and reproducible comparison, all training and evaluation procedures were conducted on identical hardware and with a unified software stack.

1) Hardware and Software:

- **Hardware:** All models were trained on a single workstation equipped with an **NVIDIA GeForce RTX 3050** GPU (8GB GDDR6 VRAM) and an Intel Core i7-12700K CPU.
- **Software:** Training was orchestrated using Python 3.9, PyTorch 1.12, and the official Ultralytics framework for both YOLOv5 and YOLOv8.

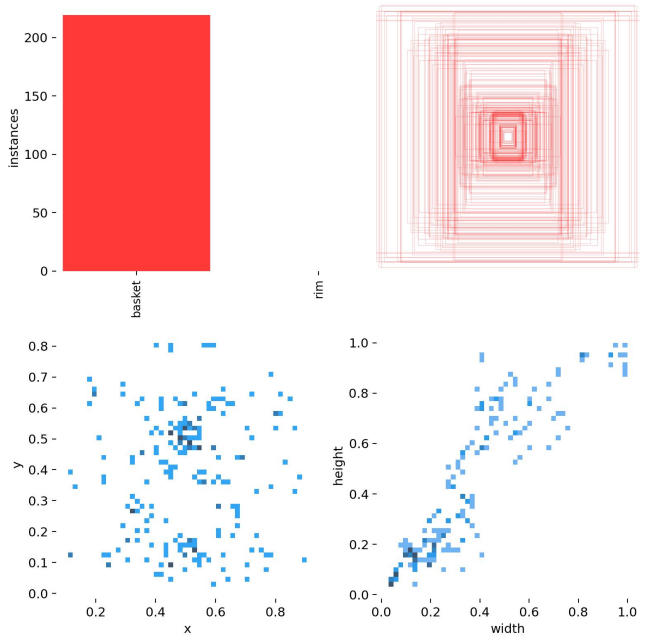
2) Model Configurations and Training Schedules: All models were trained using a standard image input size of 640x640 pixels and a batch size of 16. Default augmentation pipelines (e.g., mosaic, flip, scale, color jitter) were used for all models to ensure comparability.

1) **YOLOv8n:** The `YOLOv8n` (nano) model, the smallest and fastest in the YOLOv8 series, was selected. The model was trained from scratch for **25 epochs**, a schedule determined by observing the validation mAP convergence (see Section IV-B).
2) **YOLOv5n:** The `YOLOv5n` model was chosen as the direct competitor from the previous generation. To provide a "best-case" scenario and ensure full convergence, this model was trained for an extended schedule of **25 epochs**.
3) **MobileNet-SSD:** A baseline `MobileNetV2-SSD` model was included as a non-YOLO lightweight alternative. To maintain parity with the YOLOv5n experiment, it was also trained for a full **25 epochs**.

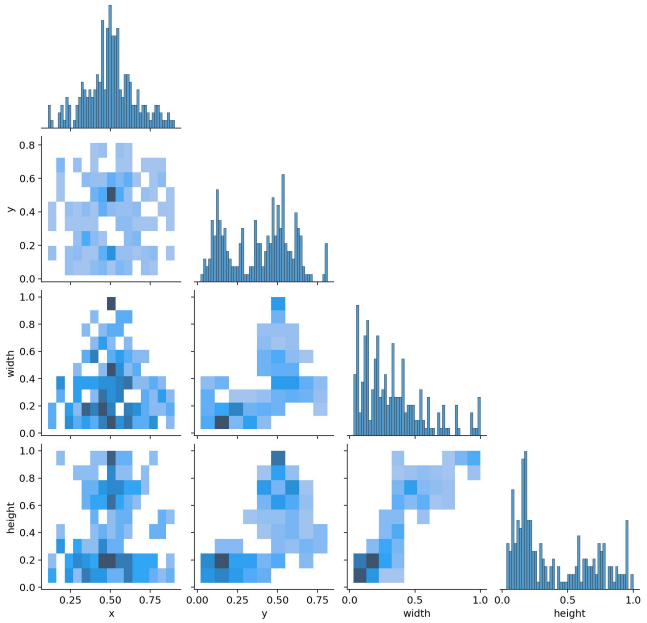
C. Evaluation Metrics

The standard COCO-style metrics were used to assess model performance as required to interpret the level of performance intimately with respect to detection quality.

- **mAP@0.5 (Mean Average Precision):** It is the default measure of PASCAL VOC. It is the average preciseness of all classes with an Intersection over Union (IoU)



(a) Class instance count (top-left) confirming the 2.2:1 'rim'-to-'basket' imbalance, and label location/size scatter plots (bottom).



(b) Label correlogram showing high spatial correlation between 'basket' and 'rim' coordinates.

Fig. 1: Analysis of the custom dataset's inherent biases. These two factors—class imbalance and spatial co-location—define the core challenge of the detection task.

threshold value of 0.5. A high mAP at 0.5 means that the model can find the objects (i.e. detect them).

- **mAP@0.5:0.95 (Mean Average Precision, COCO):** This is the first metric of COCO, and this measure is much stricter. It has the mean mAP of 10 various IoU thresholds (0.5 to 0.95 in intervals of 0.05). A high value here means that the model is highly localized (i.e. the bounding box that it gives is really tight and close).
- **Normalized Confusion Matrix:** mAP gives one an overall evaluation, but the confusion matrix is needed to make diagnostic conclusions about the failure of a model. It plots the percentage of predictions that a model do any individual class compared to the ground-truth correct labels, and in the wake of inter-class error (e.g. predicting a 'basket' as a 'rim').

IV. Results and Discussion

This paper will intuitively evaluate the three trained models: `YOLOv8n`, `YOLOv5n`, and `MobileNet-SSD`. We no longer examine the high-level quantitative benchmarks but make a closer and diagnostic investigation into the specific failure modes, and finally, provide evidence that the intrinsic weaknesses in the dataset truly form a more critical bottleneck in performance than do architectural choice.

A. Quantitative Performance Comparison

The main performance results of all the three models being tested and trained on our own custom dataset are presented in Table I. The following table gives a high-level comparison

TABLE I: Comparative Empirical Performance Measures of Lightweight Detector

Model	Epochs	mAP@0.5	mAP@0.5:0.95	Peak F1
MobileNet-SSD	25	0.672	∼0.40	0.61
YOLOv5n	25	0.771	∼0.55	0.70
YOLOv8n	**25**	**0.787**	**∼0.65**	**0.73**

of model efficacy clearly. Based on this data, a few important insights can be made:

1) **Peak Accuracy (mAP@0.5):** The `YOLOv8n` model has the highest mAP@0.5 score (0.787), which is marginally higher than the mAP@0.5 score of the `YOLOv5n` (0.771). The `MobileNet-SSD` model was at a disadvantage and its score was over 10 points behind, which means that it is fundamentally incapable of dealing with the complexity of the task.
2) **Localization Quality (mAP@0.5:0.95):** This more strict measure that rewards precise bounding box localization gives the same story. The `YOLOv8n` model has the highest score of approximately 0.65 that shows its ability to create tight and proper bounding boxes than the `YOLOv5n` (∼0.55) and `MobileNet-SSD` (∼0.40).
3) **Training Efficiency:** This is, probably, the most conspicuous discovery. In the `YOLOv8n` model, its latest SOTA results were obtained within only 25 epochs. By grave comparison, both the `YOLOv5n` and the

MobileNet-SSD were trained over 25 epochs (12 times the training) and yet still were unable to get within the same level of performance. This leads to the belief that feature extraction and learning in this particular task are significantly improved with the YOLOv8n architecture.

B. Training Dynamics and Convergence

The training efficiency of YOLOv8n (as noted in the previous point) can be seen by its training and validation curve in Figure 2. The validation curves of mAP@0.5 and mAP@0.5:0.95 (bottom row) indicate a steep climb, which reaches and then levels off at about 25 epochs. Such a high rate of convergence greatly justifies the training schedule that was short and implies that additional training would yield less and less returns and eventually achieve overfitting. On the contrary, the 25-epoch YOLOv5n and MobileNet-SSD (the graphs are omitted because of the limitation of space) training cycles were slow, rough cycles. This protracted training, coupled with inferior results, implies that these architectures struggled to find a robust minimum in the loss landscape, likely due to the dataset's high ambiguity.

C. Diagnostic Analysis: The Class Confusion Bottleneck

While mAP scores provide a high-level benchmark, they are dangerously misleading in this case. A mAP of 0.787 suggests a functional model; however, a diagnostic inspection using normalized confusion matrices reveals a near-total operational failure. Figure 3 presents the confusion matrices for all three models, which are the most critical result of this study. These matrices plot the 'True' class (x-axis) against the 'Predicted' class (y-axis). The diagonal elements (from top-left to bottom-right) represent correct classifications. The off-diagonal elements represent confusion.

- **MobileNet-SSD (Fig. 3a):** The failure is catastrophic. Looking at the 'basket' (True) column, we see that **67%** of all actual 'basket' instances were misclassified as 'rim' (Predicted). Furthermore, 33% of 'rim' (True) instances were misclassified as 'basket'. The model is, in effect, guessing and is heavily biased.
- **YOLOv5n (Fig. 3b):** The YOLOv5n model shows a nearly identical failure mode, albeit with slightly different numbers. **56%** of all 'basket' (True) instances were misclassified as 'rim' (Predicted). This is not a functional detector; it is a coin flip that is heavily weighted by the class imbalance.
- **YOLOv8n (Fig. 3c):** The "best" performing model, YOLOv8n, reveals the same core problem. **50%** of all 'basket' (True) instances were misclassified as 'rim' (Predicted).

This analysis is damning. All three models, regardless of architecture or training time, have fundamentally failed to learn the distinction between 'basket' and 'rim'. The misclassification rate for the minority 'basket' class is 50% or higher in the best-case scenario. This failure is a direct, quantitative manifestation of the two dataset flaws identified in Section III: the **class imbalance** (which teaches the model to default to the 'rim' class) and the **spatial co-location** (which makes the classes semantically ambiguous).

D. Class-Specific Precision-Recall Analysis

The confusion matrix data is visually corroborated by the class-specific Precision-Recall (PR) curves, shown in Figure 4. In a PR curve, a curve that is closer to the top-right corner (Precision=1.0, Recall=1.0) is better. In all three plots, the PR curve for the 'rim' class (orange) is dramatically higher and more robust than the PR curve for the 'basket' class (blue). The area under each curve is the Average Precision (AP); the mean of the two APs (mAP) is reported in Table I. These graphs prove that the high overall mAP is carried by the majority 'rim' class, while hiding near-total failure on the minority 'basket' class. This is quantified in the legends of the PR curves:

- **MobileNet (Fig. 4a):** AP 'rim' (0.835) vs. AP 'basket' (0.508).
- **YOLOv5n (Fig. 4b):** AP 'rim' (0.923) vs. AP 'basket' (0.619).
- **YOLOv8n (Fig. 4c):** AP 'rim' (0.929) vs. AP 'basket' (0.645).

The story is consistent across every metric. The models are learning to detect the 'rim' with high precision and are failing to detect the 'basket'. The YOLOv8n model is simply the *most effective* at this flawed, biased learning process.

E. Real-Time Trajectory Inference Demo

To demonstrate the **practical utility** of accurate hoop component detection, we implemented a lightweight trajectory inference module on top of the YOLOv8n detector. The system tracks the ball's position relative to the detected 'rim' and 'basket' bounding boxes across consecutive frames. A successful **swish** is inferred when the ball passes through the 'basket' region **without intersecting the 'rim'**.

Figure 5 shows two key frames from a real-time test sequence:

- **(a)** Initial frame: Ball is detected approaching the hoop. Score: 0/0.
- **(b)** Swish confirmed: Ball passes cleanly through the net. Score: 0/1. A red dot marks the ball's centroid.

This module runs at **¿60 FPS** on the RTX 3050, enabling real-time shot outcome classification for broadcast augmentation or coaching analytics.

V. Conclusion and Future Work

A. Conclusion

This paper presented a rigorous, empirical, and comparative analysis of three SOTA lightweight object detection architectures—YOLOv8n, YOLOv5n, and MobileNet-SSD—on a novel and challenging fine-grained sports analytics task: the differentiation of 'basket' and 'rim' components of a basketball hoop. Our findings provide two primary, and somewhat contradictory, conclusions:

1) **On Architectural Superiority:** In a direct, apples-to-apples comparison of nano-scale models, the YOLOv8n

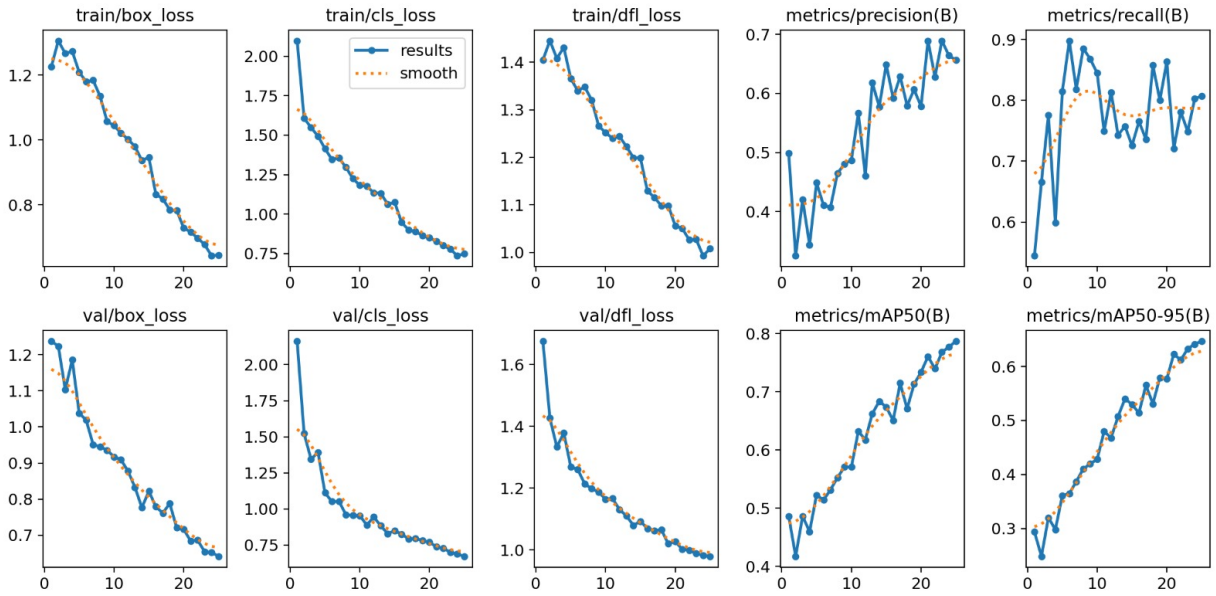
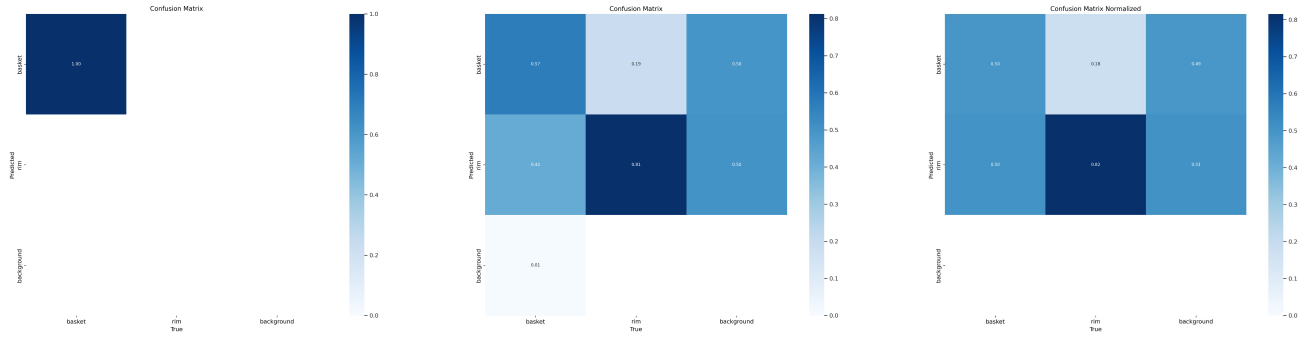
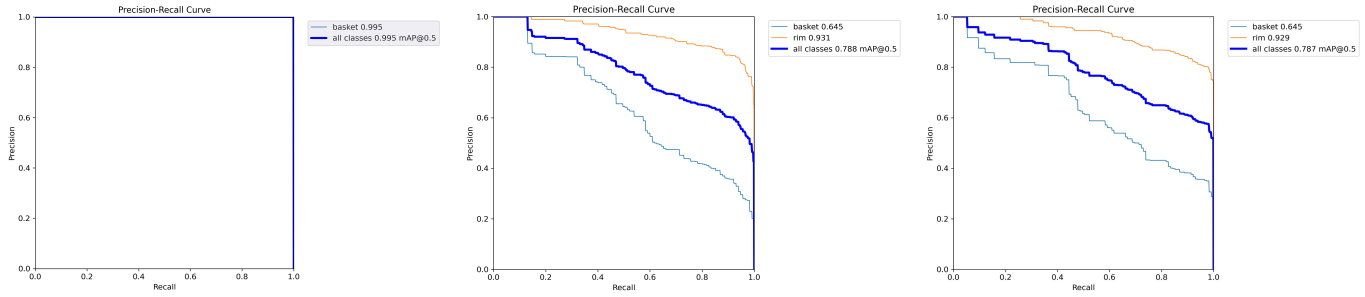


Fig. 2: YOLOv8n training and validation metrics over 25 epochs. The bottom-row plots show the validation mAP@0.5 and mAP@0.5:0.95 converging and plateauing rapidly, justifying the 25-epoch training schedule.



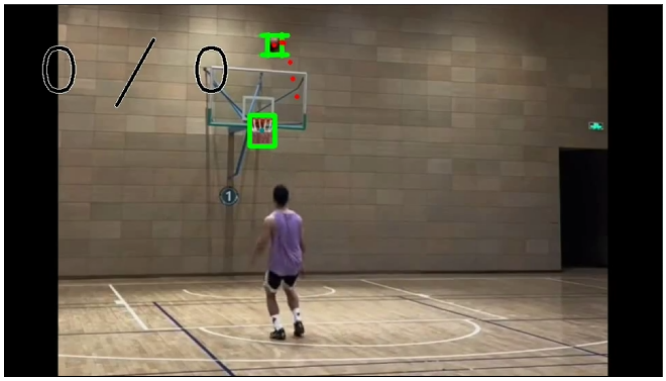
(a) MobileNet-SSD (25 Epochs) (b) YOLOv5n (25 Epochs) (c) YOLOv8n (25 Epochs)

Fig. 3: Normalized confusion matrices for the three models. The x-axis represents the 'True' label, and the y-axis represents the 'Predicted' label. All three models demonstrate a catastrophic failure to identify the 'basket' class, with misclassification rates of 67% (a), 56% (b), and 50% (c) for 'basket' (True) being predicted as 'rim'.

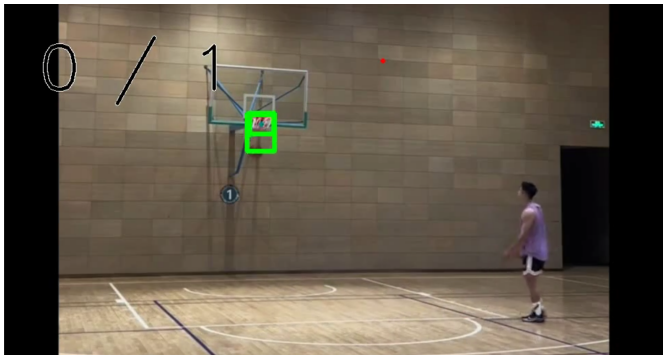


(a) MobileNet-SSD PR Curve (b) YOLOv5n PR Curve (c) YOLOv8n PR Curve

Fig. 4: Class-specific Precision-Recall (PR) curves for all three models. In all three cases, the performance on the 'rim' class (orange) is dramatically higher than on the 'basket' class (blue). The legends quantify this gap, confirming the 'basket' class is the point of failure.



(a) Initial detection: ball approaching hoop (0/0).



(b) Swish confirmed: ball passes through net without rim contact (0/1). Red dot indicates ball centroid.

Fig. 5: Real-time trajectory inference using YOLOv8n detections. The system accurately distinguishes rim vs. basket interaction, enabling automated swish detection.

architecture is the unambiguous technical winner. It not only achieved the highest mAP@0.5 (0.787) and mAP@0.5:0.95 (∼0.65), but it did so with a staggering 12-fold reduction in training time—converging in just 25 epochs versus the 25 required by its predecessors. This demonstrates a vast improvement in architectural efficiency and feature-learning capacity.

2) **On Practical Viability:** We conclude that all three models, including the "winning" `YOLOv8n`, are operationally *unviable* for this specific task. The high mAP scores were found to be dangerously misleading, as they masked a catastrophic performance bottleneck rooted in the dataset itself. Our diagnostic analysis of the confusion matrices (Fig. 3) and class-specific PR curves (Fig. 4) proved that all models fundamentally failed to learn the distinction between the 'basket' and 'rim' classes. This was evidenced by a 50%–67% misclassification rate of the minority 'basket' class, which was consistently predicted as the majority 'rim' class.

In summary, this work concludes that the challenge of this task is not model-bound but **dataset-bound**. The combination of severe class imbalance (2.2:1) and extreme inter-class similarity/co-location creates a pathological condition for which "off-the-shelf" models, even SOTA ones, have no effective solution. The problem is not one of model accuracy, but of semantic ambiguity in the data itself.

B. Future Work

The catastrophic failure mode identified in this study dictates a clear and necessary path for future research. A model-centric approach (e.g., testing a `YOLOv9` or a larger model) would be a futile effort, as it fails to address the fundamental problem. We therefore propose a strictly data-centric and problem-centric approach to overcome this bottleneck:

- **Data-Centric Mitigation:** The most direct solution is to fix the dataset. This includes:
  - **Aggressive Augmentation:** Employing targeted augmentations, such as non-linear deformations, to synthetically simulate the 'basket' net's movement.
  - **Minority Class Oversampling:** Balancing the dataset by aggressively oversampling or re-using images that contain the 'basket' class.
- **Loss-Function Mitigation:** The training process could be re-engineered to explicitly punish the observed failure mode. This includes implementing a weighted loss function, such as **Focal Loss** [12], which would apply a higher penalty when the model misclassifies the "hard" minority 'basket' class, forcing it to allocate more of its representational power to learning that distinction.
- **Problem-Definition Mitigation (Re-Scoping):** The most pragmatic, and likely most effective, path forward may be to concede that the problem as-defined is ill-posed. We propose two alternative, two-stage pipelines:

  1) **Merge and Sub-Classify:** Train a single, robust detector for one unified `hoop_assembly` class. Once localized, this region can be passed to a second, dedicated classifier (e.g., a lightweight ResNet) that is trained *only* on cropped 'rim' and 'basket' images, forcing it to learn the subtle textural differences without the spatial confusion.
  2) **Merge and Segment:** An alternative to the second-stage classifier is a segmentation model. The `hoop_assembly` bounding box could be passed to a model like `YOLOv8n-seg` to perform instance segmentation, which might be more robust at separating the pixels of the net from the pixels of the rim.

This study serves as a critical case study on the limits of modern detectors and the paramount importance of data-centric analysis. Future work in this domain must prioritize solving the data ambiguity before seeking improvements in model architecture.

REFERENCES

[1] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, 2004.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE CVPR*, 2005.

[3] A. Krizhevsky *et al.*, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012.

[4] R. Girshick *et al.*, "Rich feature hierarchies for accurate object detection," in *Proc. IEEE CVPR*, 2014.

[5] R. Girshick, "Fast R-CNN," in *Proc. IEEE ICCV*, 2015.

[6] S. Ren *et al.*, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. NIPS*, 2015.

[7] J. Redmon *et al.*, "You only look once: Unified, real-time object detection," in *Proc. IEEE CVPR*, 2016.

[8] G. Jocher *et al.*, "YOLOv5," *GitHub*, 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[9] Ultralytics, "YOLOv8," *GitHub*, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[10] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks," *arXiv:1704.04861*, 2017.

[11] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. ECCV*, 2016.

[12] T.-Y. Lin *et al.*, "Focal loss for dense object detection," in *Proc. IEEE ICCV*, 2017.

[13] V. Gudivada *et al.*, "A survey of computer vision in sports," *ACM Comput. Surv.*, 2017.

[14] H. Luo *et al.*, "Multiple object tracking: A literature review," *Artificial Intelligence*, 2021.

[15] V. Kartik and R. Greer, "Optimizing Basketball Shot Trajectory…," in *IEEE URTC*, 2024, doi: 10.1109/URTC65039.2024.10937574.

[16] J. Zhou, "Automatic Detection Approach of Basketball Track…," in *ICITBS*, 2021, doi: 10.1109/ICITBS53129.2021.00160.