YOLO-Based Deep Learning models for Real-Time Volleyball Action Recognition

Brandon Labio
The Harker School, San Jose, CA, USA
e-mail: brandonlabio@gmail.com

Atul Dubey
AIClub Research Institute, CA, USA
e-mail: atul.dubey@aiclub.world

***Abstract*—Analyzing player performance through highlights and statistics is critical for volleyball coaching, but remains a labor-intensive bottleneck. Current methods are often hindered by impractical deployment, inefficiency, and lack of portability. To overcome these limitations, this study introduces a transportable, deep-learning-based object detection model for real-time identification of volleyball actions. Using a dataset of 1,236 images, models were trained based on both YOLOv8 and YOLO11. The YOLOv8 model yielded superior performance, achieving a mAP50 value of 87.8%, and was optimized at a learning rate of 0.0005 over 80 epochs. This model was subsequently tested using the test dataset and achieved a mAP50 of 79.8%. The system was then deployed on a Raspberry Pi to evaluate real-time feasibility, reaching a latency of 1 FPS. These findings establish a critical baseline for the development of live automated highlight generation systems and are essential for efficient creation of highlights and analysis of performance.**

Keywords-Volleyball action detection; deep learning; YOLO11; YOLOv8

I. INTRODUCTION

In volleyball, understanding players' strengths, weaknesses, and playing styles is essential for coaches to support player development and plan for future matches. Players also rely on performance highlights to showcase their abilities for college recruiting [1]. Both performance analysis and highlight creation require careful review of match recordings. However, manually analyzing full-length games to extract key plays and generate statistics is inefficient and time-consuming.

In addition to requiring substantial effort, manual review is prone to human error, as important actions may be overlooked or misclassified. Therefore, there is a clear need for automated systems capable of generating highlights and match statistics accurately and efficiently. While solutions such as Balltime [2] by Hudl provide action detection and automated statistics using a proprietary computer vision model, they lack the ability to generate full game highlights, limiting their usefulness for comprehensive performance evaluation.

Our unique approach focuses on creating both highlights and statistics in real time, enabling coaches to guide players during matches and also for highlights to be ready immediately after the game for others to watch. Our model detects 6 different volleyball actions: blocking, digging, passing, serving, setting, and blocking. In addition, the device tracks the detection time, which helps generate highlights for specific actions and statistics on the frequency of different actions.

To achieve this, transfer learning was performed on pre-trained YOLOv8 and YOLO11 models to identify different actions from a recording of a volleyball game. Their performance was compared with respect to the mAP50 value and latency during inference on a resource-constrained device like Raspberry Pi. The device we built provides immediate access to game insights and supports better decision-making.

The rest of the paper is structured as follows: Section 2 reviews related works in similar domains and contrasts them with our approach. Section 3 describes the tools, techniques, and methods used to conduct this research. Section 4 presents the results, followed by a discussion of findings and potential directions for future work.

II. RELATED WORKS

In the domain of action recognition, different techniques and models have been explored, which have achieved some levels of success for various datasets.

A. Convolutional Neural Network-based approach

In [3], for real-time action detection, CNNs have been employed to decompose complex human activities into three distinct levels, yielding mean average precision scores of 76.6% for frame-based and 83.5% for video-based evaluations on the ICVL (Intelligent Computer Vision and Learning Lab) Video Surveillance Dataset. This approach highlights the value of breaking down complex actions for enhanced representation and detection.

In another study, researchers explored multi-sport action recognition using methods such as 3D CNNs, two-stream approaches, and SlowFast. These techniques were tested against a dataset featuring 3,200 video clips with annotated actions, achieving a frame-based mAP of 27.72% and varying video-based mAPs. Such results emphasize the capability of these models to handle complex, spatio-temporally localized actions [4].

In [5], researchers demonstrated the use of Kinect and CNN classifiers to interpret geometric limb relations. This method achieved an average recognition rate of 93% across several datasets, showcasing the potential of combining skeletal data with deep learning for improved action identification.

For volleyball action recognition, dual-stream CNNs were employed within the DNet model to process both spatial and temporal data. This approach led to classification accuracies of 94.12% on the UCF101 dataset, illustrating how combining different data streams can enhance model performance [6]. Another study focused on applying Gaussian Mixture Models with multi-resolution 3D CNNs to recognize movements from volleyball game videos, improving accuracy by 3.3% while reducing complexity by 33.6% [7].

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

B. YOLO based approach

Volleyball setting tactic detection has benefited from the use of YOLO-based player detection and multi-object tracking systems. These were applied to RGB images from national team matches to classify set types, resulting in improved accuracies under both clean and noisy conditions [8]. This demonstrates the effectiveness of integrating object detection and tracking with action classification.

C. CNN with vision transformer-based approach

Meanwhile, a study leveraging CNNs and Transformers, such as VideoMAE with ViT-B, focused on detecting human interactions in sports. This method utilized spatio-temporal context reasoning, achieving mAP scores for classification at 10.69% and detection at 4.93%, which highlights the complexity of modeling interactions within dynamic environments [9].

D. Long Short-Term Memory-based approach

Lastly, for sports motion recognition in the MARS dataset, the combination of millimeter-wave radar with Kalman filtering and LSTMs demonstrated robust results. By producing 5D point clouds and reconstructing models to capture temporal features, the approach attained accuracy rates of 95.6% outdoors and 97.9% indoors, underscoring its effectiveness for precise action recognition [10]. Collectively, these studies reflect the diverse approaches and outcomes in the evolving field of sports action recognition.

E. Our approach - YOLO model deployed in an edge device

Building upon the limitations observed in previous works, this project addresses key challenges in efficiency and practical deployment within volleyball action recognition. Previous studies using dual-stream CNNs and GMM-based 3D CNNs [6][7] achieved solid accuracy but relied on computationally heavy models that limited real-time application. Similarly, YOLO-based detection systems [8] improved volleyball setting recognition but did not provide automated data logging or real-time feedback. The firmware used in this project integrates a lightweight YOLOv8-based detection pipeline capable of on-device inference, enabling efficient, real-time processing of volleyball actions. Additionally, in contrast to earlier CNN- and Kinect-based methods [3][4][5], this system enhances data management and evaluation consistency by timestamping detections, saving results automatically to CSV files, and overlaying detections on live video feeds. Furthermore, by exposing both the live feed and detection data over an HTTP connection, the project provides a practical framework that bridges the gap between research-oriented recognition models and real-world volleyball analytics applications.

III. Material and Methods

The following describes the dataset, model architecture, training procedure, and hardware implementation used to develop and evaluate the volleyball action detection system.

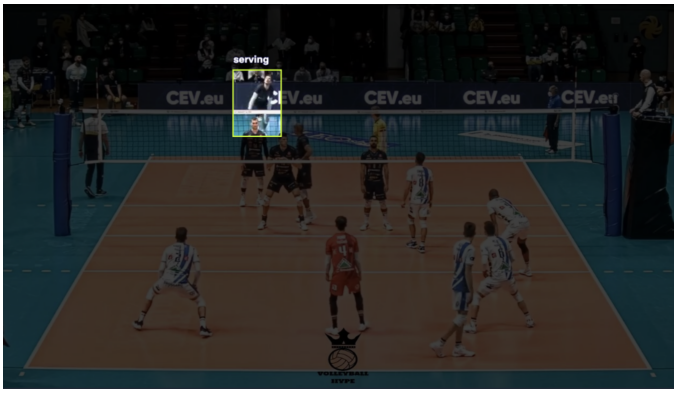

Figure 1. Example of Image From Dataset

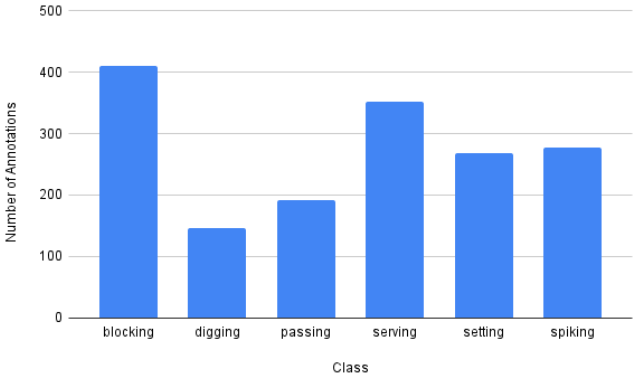

Figure 2. Distribution of annotations across images

A. Dataset

The dataset used for building deep learning models has been taken from an open source repository in Roboflow Universe [11]. It contains 1,236 images and 1646 total annotations from the same match, with there being 6 different types of annotations: blocking, digging, passing, serving, setting, and spiking. An example of an image from the dataset is presented in Figure 1. The overall distribution of annotations across images is presented in Figure 2. There is no need for negative examples since that would primarily be needed for detecting "no action," but actions are present in every frame.

B. Deep Learning models

1) YOLOv8 [12]: YOLOv8 is an efficient and lightweight object detection model, capable of real-time detection. YOLOv8 demonstrates enhanced accuracy over its predecessors by incorporating advancements in feature extraction and model architecture, resulting in better precision and recall. We have chosen to experiment with this model due to its real-time detection capability and ability to deploy on an edge device.

2) YOLO11 [13]: YOLO11 achieves a higher mean Average Precision (mAP) on the COCO dataset while using 22% fewer parameters than YOLOv8, making it computationally efficient without compromising accuracy. We chose this as it is the latest model in the YOLO series and is considered to be an improvement over YOLOv8.

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

C. Model Training

As we can see in Figure 3, before the start of the experiment, the dataset was split into 3 different parts: train (936), validation (150), and test (150). The respective splits were around 75%, 12.5%, 12.5%, combining the common 80/10/10 and 70/15/15 strategies, designed to balance the need for sufficient training data with the need for reliable, unbiased performance estimation. After the splitting, the dataset was exported from Roboflow in YOLOv8 format. The Ultralytics Python library with Google Colab was used for performing transfer learning on the YOLOv8 and YOLO11 models to customize them for detecting volleyball actions. Experiments were conducted by varying the number of epochs from 50 to 100 and adjusting the learning rates between 0.00001 and 0.01. These hyperparameters were chosen as a starting point. However, during the training, we realized that even with lower learning rates and higher epochs, there would not be much improvement in performance. The best-performing model was subsequently evaluated using the test dataset. The code for the project is available for reference in [14].

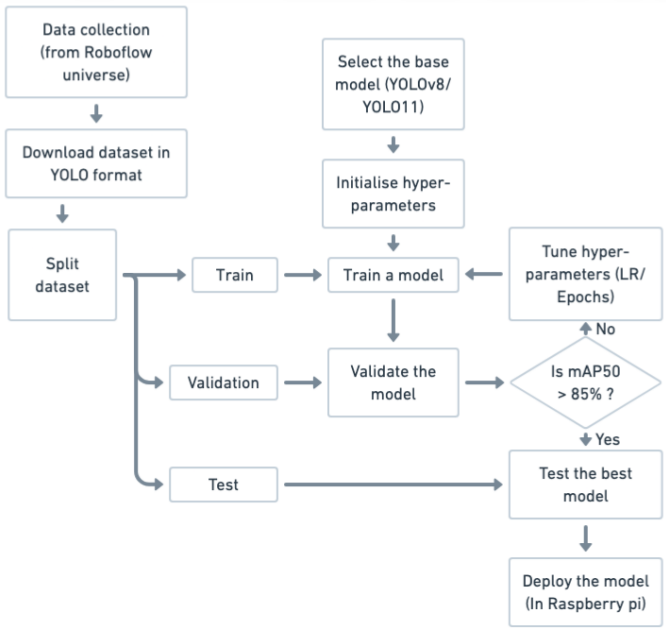

Figure 3. Machine Learning Pipeline

D. Hardware

A Raspberry Pi and a Logitech Brio 101 Full HD 1080p camera were used for testing the best model. The Raspberry Pi is a low-cost single-board computer with enough computational power to run an object detection model on the device, and the Logitech Brio 101 Full HD 1080p camera helped to capture video frames for input. The power supply provided a stable 5V/3A power source to the Raspberry Pi, ensuring reliable operation.

E. Hardware Architecture

The device consists of a Raspberry Pi 5 module, a USB camera, and a power source (Currently a power adapter, but can be changed to a powerbank for a portable setup). The camera connects to one of the USB ports of the Raspberry Pi and is powered through the port itself. Figure 4 shows connections between various components of the device.

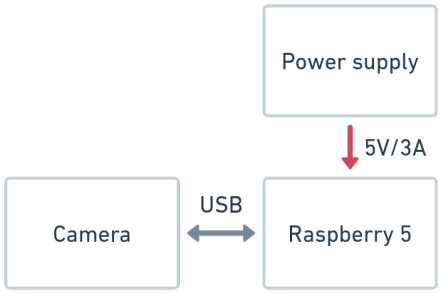

Figure 4. Hardware architecture diagram

F. Firmware

In the firmware of the device, first, the camera is initialized. After that, the YOLO11 model is loaded using the Ultralytics Python library. The camera begins capturing frames, which are subsequently processed by the model for detection. Detected actions are recorded in a CSV file along with a timestamp. A web server is implemented to enable viewing of the live feed and detection results over a local IP address. The web portal, hosted by the Raspberry Pi, displays both the live stream and a table of previous detections. Figure 5 illustrates the logic flow incorporated in the firmware.

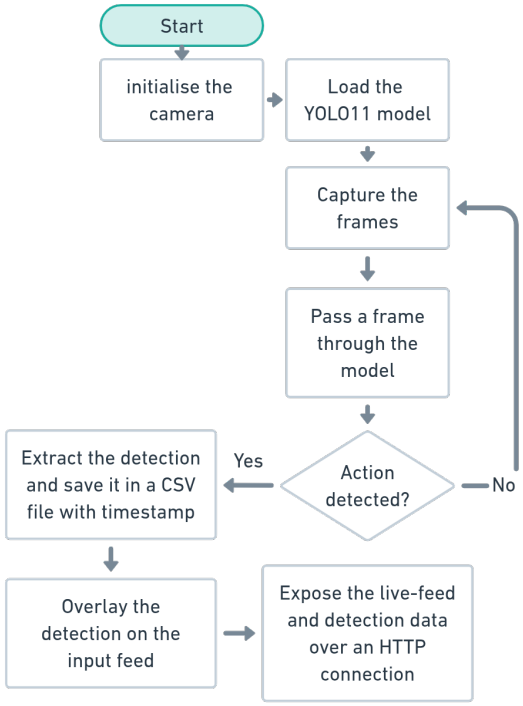

Figure 5. Firmware flowchart

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

IV. Results

The following presents experimental findings for YOLOv8 and YOLO11 models, highlighting validation performance, convergence behavior, and detection accuracy across multiple evaluation metrics.

A. YOLOv8

A total of 30 experiments were performed by varying epochs between 50 and 100 and learning rates between 0.00001 and 0.01. The best validation mAP50 of 87.8% was achieved at a learning rate of 0.0005 with 80 epochs. Figure 6 represents the variation of mAP50 values on the validation set across different experiments performed.

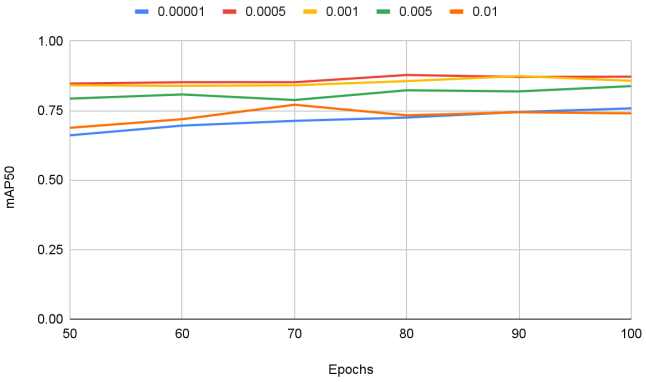

Figure 6. YOLOv8 - mAP50 vs epochs for various learning rates

Figure 7 shows the validation losses and evaluation metrics during training of a YOLOv8 model.

The following metrics were used to assess the performance of the detection model:

- **Box loss:** Box loss measures how accurately a predicted bounding box aligns with the ground-truth object location. It penalizes discrepancies in box coordinates—typically width, height, and center position—and encourages the model to produce tighter, well-localized boxes.
- **classification loss:** Classification loss evaluates how well the model assigns the correct class label to each detected object. It penalizes incorrect or uncertain predictions of object categories, guiding the network to improve its discriminative capability.
- **Distribution Focal Loss:** Distribution Focal Loss (DFL) is a localization loss used in modern object detectors to improve the precision of bounding-box regression. Instead of predicting a single continuous box value, DFL models each box coordinate as a discrete probability distribution over a set of bins. The loss encourages the predicted distribution to place higher probability mass near the true target location, enabling finer-grained and more accurate localization. This formulation reduces quantization error and enhances the model's ability to learn high-resolution bounding-box boundaries, particularly when combined with IoU-based localization losses.
- **mAP50:** mAP50 refers to the mean Average Precision computed at an Intersection over Union (IoU) threshold of 0.50. It measures how well the detector identifies objects and aligns predicted boxes with ground-truth boxes. A prediction is considered correct if IoU > 0.50.
- **mAP50-95:** mAP50–95 is the mean Average Precision computed over ten IoU thresholds, from 0.50 to 0.95 in steps of 0.05. This metric provides a comprehensive evaluation of detection performance by assessing both coarse and highly precise localization. Compared to mAP50, it is significantly more stringent, rewarding models that produce accurate bounding boxes across varying levels of overlap.

The box loss, classification loss, and DFL (Distribution Focal Loss) all decrease sharply in the early epochs and then stabilize, indicating the model is learning to localize and classify objects effectively. The mAP@0.5 metric increases rapidly within the first 20 epochs and then plateaus around 0.85–0.9, showing strong detection performance at a lenient IoU threshold. Similarly, the stricter mAP@0.5–0.95 rises steadily and stabilizes near 0.6, reflecting good overall precision-recall tradeoff across IoU thresholds.

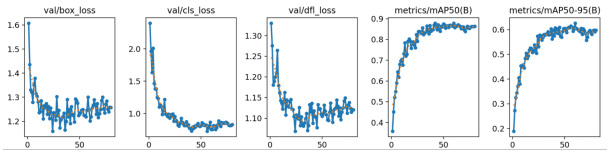

Figure 7. YOLOv8 - (a)variation of box-loss across epochs, (b) variation of class loss across epochs, (c) variation of distributed focal loss across epochs, (d)variation of mAP50 across epochs, and (e) variation of mAP50-95 across epochs

B. YOLO11

A total of 30 experiments were performed by varying epochs between 50-100 and nd learning rates between 0.00001 - and0.01. The best validation mAP50 of 87.1% was achieved at a learning rate of 0.0005 with 60 epochs. Figure 8 represents the variation of mAP50 values on the validation set across different experiments performed.

Figure 9 shows the validation losses and evaluation metrics during training of a YOLO11 model. The model converged

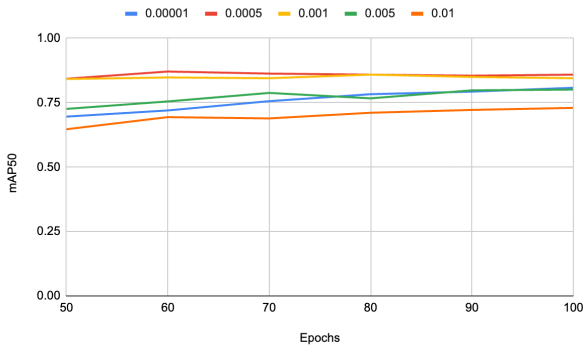

Figure 8. YOLO11 - mAP50 vs epochs for various learning rates

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

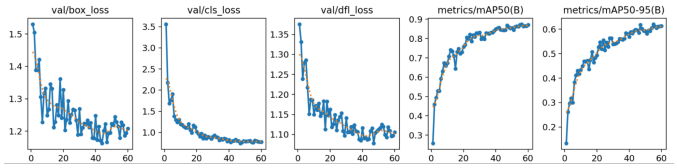

Figure 9. YOLO11 - (a)variation of box-loss across epochs, (b) variation of class loss across epochs, (c) variation of distributed focal loss across epochs, (d)variation of mAP50 across epochs, and (e) variation of mAP50-95 across epochs

well, as shown by a steady decline in validation losses and the corresponding rise in accuracy metrics. Specifically, the box loss, classification loss, and distribution focal loss decreased notably over the epochs, indicating better bounding box alignment and class prediction. The detection accuracy showed significant gains, with mAP50 increasing from 0.25 to approximately 0.88 and the stricter mAP50-95 rising from 0.15 to about 0.62.

C. Summary and Test Results

TABLE I. VALIDATION RESULT SUMMARY

Model	Best mAP50 (%)
YOLOv8	87.8
YOLO11	87.1

Among the models presented in Table I, YOLOv8 demonstrated the highest performance and was subsequently evaluated using the test dataset, achieving a mAP50 of 79.8%. Table II shows the detailed test results for each of the detected classes.

TABLE II. TEST RESULTS FOR THE BEST YOLOV8 MODEL

Class	Box Precision	Box Recall	mAP@50	mAP@50-95
blocking	0.968	0.871	0.925	0.682
digging	0.667	0.462	0.543	0.418
passing	0.773	0.548	0.643	0.490
serving	1.000	0.972	0.986	0.689
setting	0.800	0.800	0.819	0.526
spiking	0.964	0.750	0.870	0.594
Average	**0.862**	**0.734**	**0.798**	**0.567**

Figure 10 shows the Precision–Recall (PR) curve for the YOLOv8 model on the test set.

Precision and recall help reveal different types of detection errors: low precision indicates many false positives, meaning the model is predicting objects that are not present or assigning the wrong class, while low recall indicates many false negatives, where the model fails to detect objects that actually exist. Mispredictions contribute to both errors—counting as a false positive for the incorrect class and a false negative for the true class—thereby reducing both precision and recall.

Figure 10 illustrates how well the model balances precision and recall across different volleyball actions. The results show that the model performs exceptionally well on serving and blocking, with both achieving very high precision and recall. Spiking and setting also perform strongly, though with slightly lower precision at higher recall levels. In contrast, passing and especially digging (0.543) are more challenging, with lower overall precision-recall areas, suggesting room for improvement in detecting these actions.

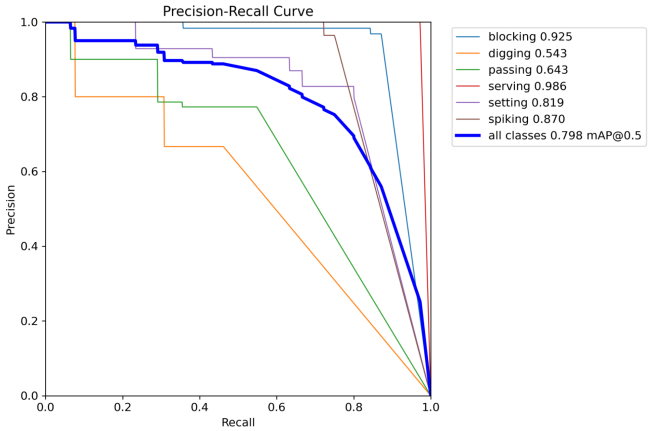

Figure 10. Precision-recall curve for the test results obtained from the best YOLOv8 model

D. Device Test Results

The device with the best model deployed was tested on simulated scenarios as well as on the recorded videos of the game. Currently, we're able to achieve a latency of 1fps with the PyTorch model deployed using the Ultralytics library.

V. DISCUSSION

The comparative analysis of YOLOv8 and YOLO11 performance across varying learning rates reveals nuanced differences in their training efficiency and final performance. In general, a large learning rate causes the model to skip the global minimum, resulting in lower accuracy, and a very low learning rate requires more epochs to reach the global minimum. According to Figures 6 and 7, the YOLOv8 model achieves its peak mAP50 of 0.878 with a learning rate of 0.001 at 80 epochs, and its performance at the same learning rate remains strong across the training duration. In contrast, the YOLO11 model reaches a similar peak mAP50 of 0.871 with a learning rate of 0.0005, but it achieves this peak significantly earlier, at just 60 epochs. This suggests that while both models can reach a comparable level of high performance, YOLO11 converges more rapidly to its optimal solution. Furthermore, the YOLO11 model's performance with a learning rate of 0.0005 shows greater stability from epoch 60 onwards, whereas the YOLOv8's performance at its peak learning rate of 0.001 shows a slight decrease after reaching its highest point at epoch 90.

As evident from Table II, the best YOLOv8 model exhibits a varied performance across different classes. The model excels at detecting 'serving', achieving a perfect box precision of 1.0 and a recall of 0.972, leading to a near-perfect mAP50 of

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

0.986. It also performs exceptionally well for 'blocking' and 'spiking', with mAP50 scores of 0.925 and 0.87, respectively, indicating a high level of accuracy for these actions. However, the model shows a notable drop in performance for 'digging' and 'passing', with mAP50 values of 0.543 and 0.643. This reduced performance is likely due to lower recall scores (0.462 for digging and 0.548 for passing), suggesting the model struggles to identify all instances of these classes. The overall average mAP50 for the model is 0.798, which, while respectable, is pulled down by the poor performance on these specific classes. These findings highlight the need for further model refinement, perhaps through additional data augmentation or architectural modifications, to improve detection of the less-represented or more challenging classes.

Furthermore, several challenges were encountered during the testing process. Establishing stable connections between the YOLO models, live camera input, and the HTTP server also posed difficulties, as latency and inconsistent frame rates occasionally disrupted real-time detection. Finally, the frame-by-frame processing approach, while efficient for isolated detections, limited temporal context awareness and occasionally led to misclassifications when consecutive actions appeared visually similar. These challenges highlight potential directions for improvement, with the most prominent next step being the incorporation of temporal modeling techniques to improve future iterations.

VI. Conclusion and Future Work

This study confirms that deep learning-based object detection is a highly effective approach for analyzing volleyball actions. Our optimized YOLOv8 model achieved an mAP50 value of 0.798, which is an acceptable result for action detection. While the current deployment on Raspberry Pi delivers accurate recognition, minimizing latency remains a priority. Future iterations will address this by converting the model to more compact formats (quantization) or utilizing external edge-TPU accelerators. Additionally, we aim to enhance the system's analytical depth by incorporating spatio-temporal features via LSTM networks and developing an ensemble architecture. This will enable simultaneous action recognition and specific player identification, paving the way for fully automated and comprehensive match reporting.

Furthermore, the current model is limited to detecting only actions but not identifying the players who did the actions. In the future, we can create an ensemble of two models where we can identify the players as well as the actions done by them. This can significantly improve the quality of reports and highlights generated.

References

[1] B. Schunzel, *Video in the Recruiting Process: Simple and Significant | Junior Volleyball Association — jvavolleyball.org*, https://jvavolleyball.org/video-recruiting-process-simple-significant/, [Accessed 12-10-2025].

[2] *Balltime from Hudl is the leading AI Volleyball Solution • Hudl — hudl.com*, https://www.hudl.com/en_gb/products/balltime, [Accessed 10-11-2025].

[3] C.-B. Jin, S. Li, and H. Kim, "Real-time action detection in video surveillance using sub-action descriptor with multi-cnn," *arXiv preprint arXiv:1710.03383*, 2017, [Accessed 12-10-2025].

[4] Y. Li et al., "Multisports: A multi-person video dataset of spatio-temporally localized sports actions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, [Accessed 12-10-2025], 2021, pp. 13 536–13 545.

[5] J. Tang, "An action recognition method for volleyball players using deep learning," *Scientific Programming*, vol. 2021, no. 1, p. 3 934 443, 2021, [Accessed 12-10-2025].

[6] B. Li and M. Tian, "Volleyball movement standardization recognition model based on convolutional neural network," *Computational Intelligence and Neuroscience*, vol. 2023, no. 1, p. 6 116 144, 2023, [Accessed 12-10-2025].

[7] H. Wang, X. Jin, T. Zhang, and J. Wang, "Convolutional neural network-based recognition method for volleyball movements," *Heliyon*, vol. 9, no. 8, 2023, [Accessed 12-10-2025].

[8] H. Xia et al., "Advanced volleyball stats for all levels: Automatic setting tactic detection and classification with a single camera," in *2023 IEEE International Conference on Data Mining Workshops (ICDMW)*, [Accessed 12-10-2025], IEEE, 2023, pp. 1407–1416.

[9] T. Wu, R. He, G. Wu, and L. Wang, "Sportshhi: A dataset for human-human interaction detection in sports videos," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, [Accessed 12-10-2025], 2024, pp. 18 537–18 546.

[10] Z. Zhang, Y. Tian, and J. Qi, "Volleyball technical action recognition based on cnn-lstm," *Discover Artificial Intelligence*, vol. 5, no. 1, p. 188, 2025, [Accessed 12-10-2025].

[11] *Volleyball_dataset Model by MIKHAIL KLYUKIN — universe.roboflow.com*, https://universe.roboflow.com/mikhail-klyukin/volleyball_dataset/images/KZi1uOP1ZvEcztaMm7Ed, [Accessed 10-11-2025].

[12] R. Varghese and M. Sambath, "Yolov8: A novel object detection algorithm with enhanced performance and robustness," in *2024 International conference on advances in data engineering and intelligent computing systems (ADICS)*, [Accessed 12-10-2025], IEEE, 2024, pp. 1–6.

[13] R. Khanam and M. Hussain, "Yolov11: An overview of the key architectural enhancements," *arXiv preprint arXiv:2410.17725*, 2024, [Accessed 12-10-2025].

[14] B. Labio, *GitHub - brandonl-byte/VolleyballActionDetection — github.com*, https://github.com/brandonl-byte/VolleyballActionDetection, [Accessed 12-10-2025], 2025.

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org