Detecting a ball in volleyball game using YOLO models

Jan Halas

September 17, 2025

Motivation and Goals

Television broadcasters, sports scouts, and data scientists increasingly rely on computer vision to enhance live streams and extract valuable insights. For example, in television broadcasts, computer vision is used to highlight the trajectory of the ball or determine the precise position of a pass without the need for manual annotation. Achieving this requires advanced computer vision methods. The goal of this project is therefore straightforward: to train a model that detects ball as first step to any computer vision project while working with a limited dataset and resources.

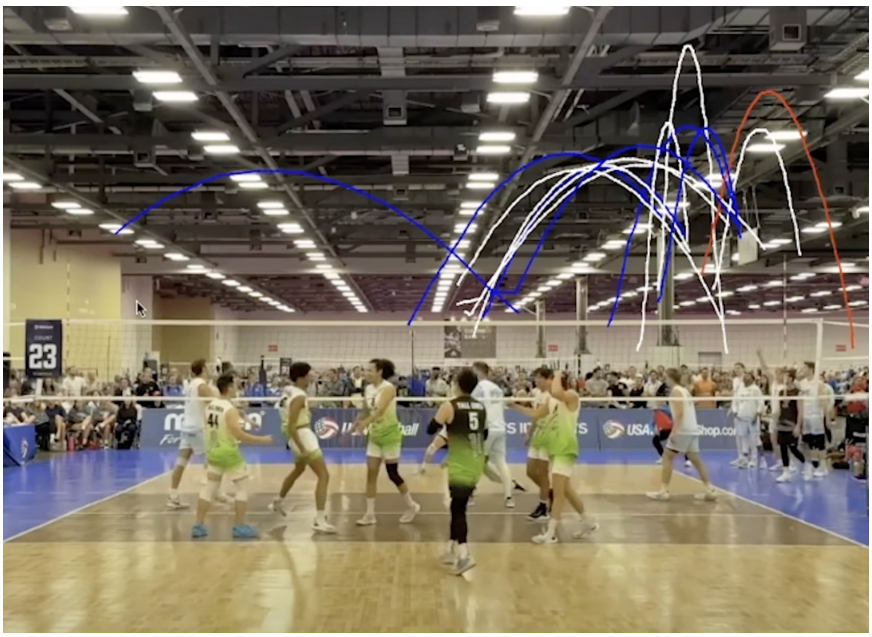

Figure 1: Example of analysing sets using AI [1].

Model Choice

To address this task, I selected the YOLOv11n [3] and YOLOv11s models, which uses 2.6M and 9.4M parameters respectively. My choice is primarily based on two considerations.

First, the nature of the problem involves detecting only a single object, the ball. Since the ball is a simple, round object that does not change shape, the task should not require the complexity of a larger model. Second, the available computational resources are limited. Training larger models would be computationally expensive and time-consuming, without providing significant benefits given the relatively small dataset.

Dataset

For this project, I collected and annotated a custom dataset. The images were sourced from matches of my hometown team and consist of 636 samples across five games. Annotation was performed using the LabelStudio [4]. The images are provided in two resolutions: 1280×720 and 1080×1920. Since the quality of the dataset is critical let's examine few examples.

Figure 2: Image example in dataset.

The images in the dataset vary in quality. In many cases, the ball occupies only a very small portion of the frame, sometimes as little as 12 pixels in height, which makes it challenging for YOLO to reliably learn its features. This issue is further compounded by the high resolution of the images. YOLO preprocesses its inputs to match the parameter `imgsz` by resizing the larger dimension to the specified value (while preserving aspect ratio) and padding the rest with gray pixels, a technique known as letterboxing. As a result, the ball object becomes even smaller. To mitigate this, the input size was set to imgsz = 736. This value represents a compromise between 1280 (which would require more computational resources) and 640 (which is default value). Additionally, an alternative approach was tested by training the model on cropped images, with results from this experiment presented later in the report.

Figure 3: Ball in top right corner is only few pixels long.

.

Training

I trained two models: YOLOv11n and YOLOv11s. The training was performed on a MacBook M1 Pro 32 GB RAM. The nano model was trained for 500 epochs, while the small model was trained for 300 epochs. In total, the training process for both models took approximately five hours.

Hyperparameters

The AdamW optimizer was selected for training, with a learning rate of 0.002. The input size parameter `imgsz` was set to 736 to partially compensate for the small size of the ball in the images. In addition, data augmentation parameters were slightly adjusted to better suit the task:

- `fliplr = 0.5` – probability of horizontal flip, helps with generalization
- `shear = 0` – disabled since the ball maintains the same shape from any viewpoint
- `scale = 0.3` – lowered to prevent the ball from shrinking further
- `bgr = 0.1` – probability of color inversion reduced, as it was not very useful
- `mosaic = 1.0` – kept as recommended by Ultralytics for detection of small objects
- `degrees = 20` – rotation up to 20° found beneficial

Training Loss

In figure 9, multiple training metrics are presented. Both precision and recall converge rapidly after approximately 100 epochs. The high precision is expected, given that the task involves detecting only a single class. Surprisingly, recall also reaches a relatively high level despite the limited dataset size. The box loss, however, does not fully converge. Nevertheless, since this is a detection task, precise bounding box alignment is less critical. Moreover, extending the training beyond the current setup does not appear to yield significant improvements in recall.

Let us further examine recall. In figure 5, the Recall–Confidence curve is shown. Recall remains high until relatively large confidence thresholds, only dropping at higher values. This

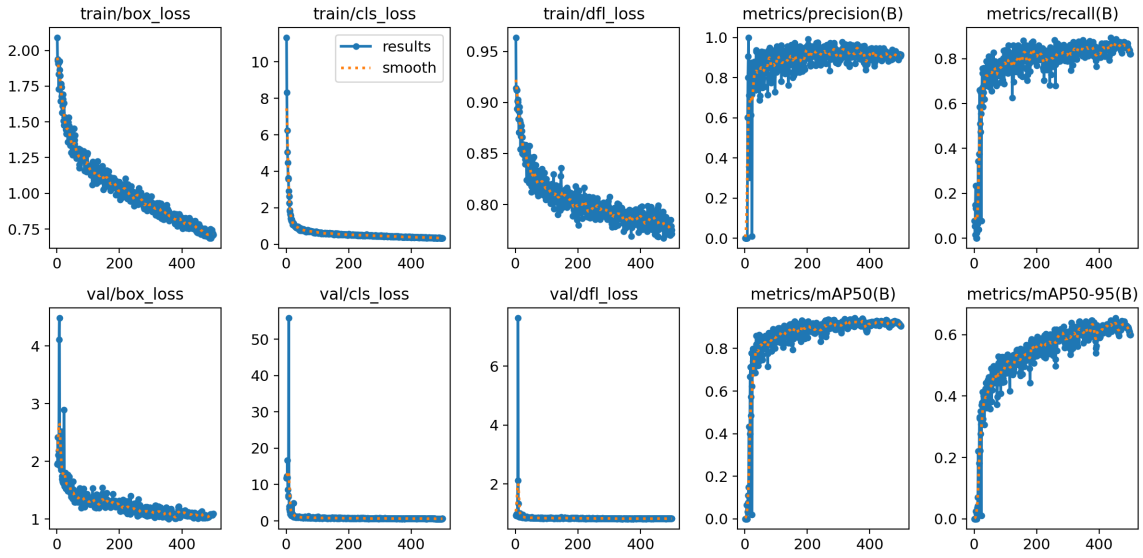

Figure 4: Metrics during training of model YOLOv11n

.

behaviour indicates that the model is able to detect the ball reliably with good confidence, resulting in robust detection quality.

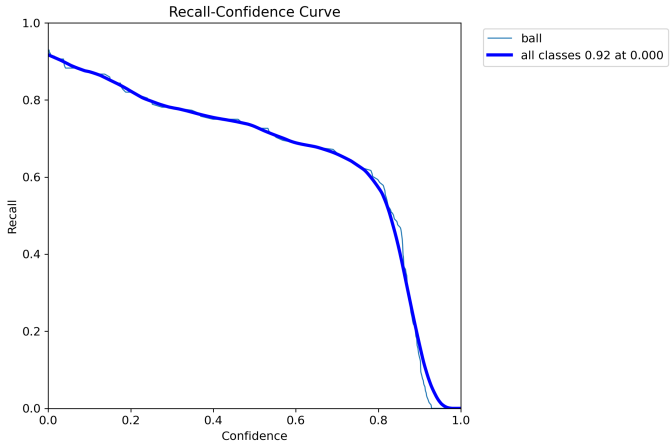

Figure 5: Recall-Confidence curve for model YOLOv11n

.

Results

Let us now examine the final results. It is evident that the nano model outperforms the small model significantly. This can be explained as the larger model is too complex for the limited dataset, whereas the smaller model is able to learn the features better. The superior performance of YOLOv11n also has practical advantages, including faster inference times, which make it more suitable for potential real-time applications.

Model	Precision	Recall
YOLOv11n	**0.96**	**0.84**
YOLOv11s	0.88	0.76

Table 1: Precision and Recall comparison for YOLOv11 models.

SAHI

SAHI [2] is a framework designed to enhance detection or segmentation of small objects. SAHI divides an image into multiple overlapping subimages that cover the entire image. Overlapping ensures that objects at the borders of subimages are not missed. Detection is then performed on each subimage individually, and any duplicate detections of the same object are resolved in a final aggregation step. This approach benefits from processing smaller fragments of the original image, which reduces the degree of shrinking applied to small objects during preprocessing.

The experiment was conducted in two parts. First, SAHI was applied to the previously trained models to evaluate whether they could effectively utilize their learned features on smaller inputs derived from the original images. Second, a new YOLOv11n model was trained on a dataset tailored for the SAHI scenario. The original images were cropped to a size of 640×640 pixels, which was also used as the parameter for SAHI slices. Cropping was performed such that the ball was always included, and the offset of each crop was selected randomly. This model was then applied via SAHI for inference on the original, non-cropped images. The intuition behind this approach is that training the model on slices rather than full-size images allows the learned features to correspond more closely to the smaller regions processed by SAHI, potentially improving detection of the ball.

Results

First, let us briefly comment on the performance of the model itself. This model was trained for 400 epochs with an input size of 640, while all other hyperparameters remained unchanged. The training metrics evolved almost identically to the previous models: precision and recall converged relatively quickly, while there is still some room for improvement in bounding box alignment, which is not critical for this task. A notable improvement can be seen in the Recall–Confidence curve (figure 10), where recall remains more stable at higher confidence thresholds compared to the previous models. Figures for small model all included in Appendix as the results are identical.

The model's performance improved significantly, which I explain as the better quality dataset was used for training. Table **??** provides a comparison with the previous models; however, this comparison is intended for illustrative purposes only, as the datasets differ.

Next, we examine the results obtained using SAHI. In this experiment, each model was evaluated on three different slice sizes: 640, 736, and 832, except for the model trained on the cropped dataset, which was evaluated only on slices of size 640 (the size it was trained on). The overlap ratio was set to 0.15, ensuring that the ball would always fit within a slice if necessary. The experiment was conducted on the original validation dataset.

The results show a clear improvement. Specifically, YOLOv11n with slice size 640

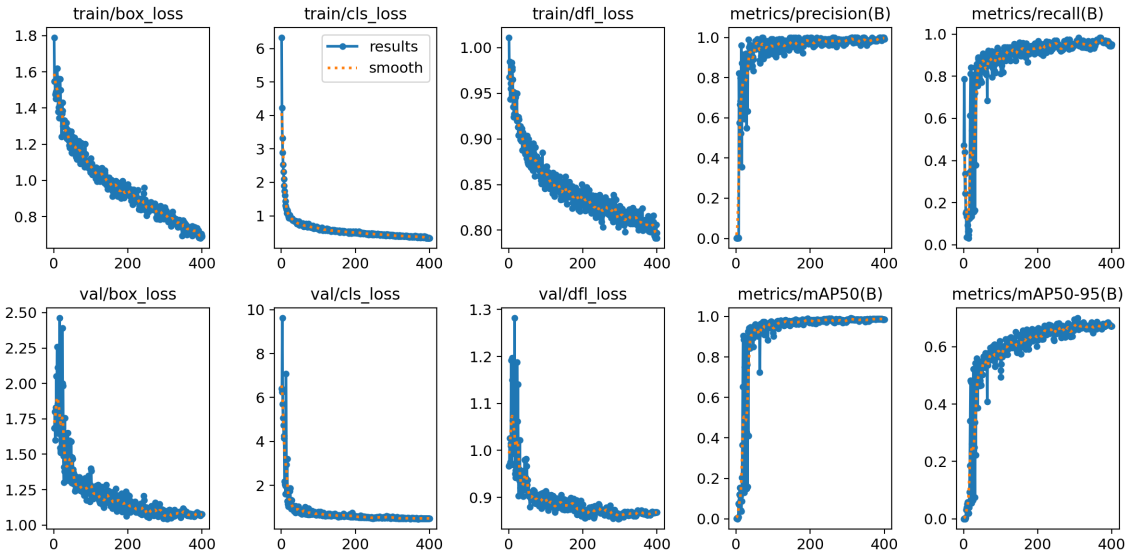

Figure 6: Metrics during training of model YOLOv11n on sliced dataset

.

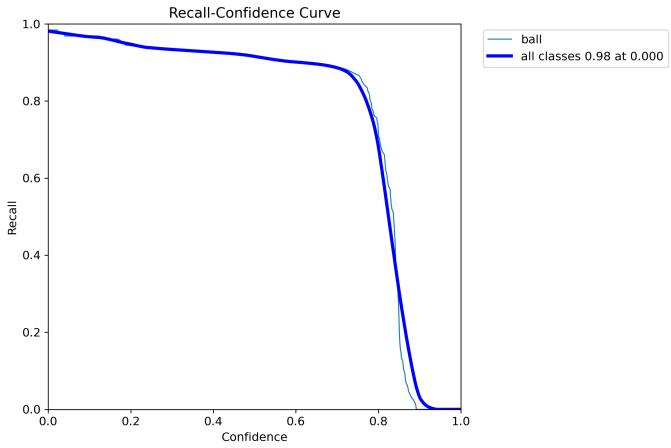

Figure 7: Recall-Confidence curve for model YOLOv11n trained on sliced dataset

.

Model	Precision	Recall
YOLOv11n sliced	**0.98**	**0.96**
YOLOv11n	0.96	0.84
YOLOv11s	0.88	0.76

Table 2: Precision and Recall comparison for YOLOv11 models.

achieved a 4 percentage point increase in recall compared to its original model. However, this improvement comes at the cost of a significant drop in precision, as the model also detects more background objects, as illustrated in figure 8. Surprisingly, the model trained on cropped slices performed 18 percentage points worse compared to its performance on its own

validation set. This behaviour is unexpected, as the slices provided by SAHI were intended to match the features of its training dataset.

Model	Precision	Recall
YOLOv11n 640	0.54	**0.86**
YOLOv11n 736	0.75	0.85
YOLOv11n 832	**0.80**	0.85
YOLOv11s 640	0.48	0.81
YOLOv11s 736	0.73	0.78
YOLOv11s 832	0.75	0.83
YOLOv11n 640 sliced	0.69	0.78

Table 3: Precision and Recall comparison for YOLOv11 models using SAHI framework.

Figure 8: SAHI predicts as ball part of jersey and ear.

.

Conclusion and Future Work

In this project three YOLO models were trained and evaluated in two scenarios. I am perosnally surprised with the results. I am convinced that with more quality dataset could be achieved with better results. Surprisingly, the results of the model trained on the sliced dataset were relatively poor and unexpectedly low.

To develop a practical, real-world application based on these models, further improvements are necessary. This includes collecting more data, training and experimenting iteratively. Perfect recall may never be achievable, so it is also important to implement techniques to handle missed detections, such as techniques for interpolating the coordinates of objects in frames where detections are absent.

Appendix

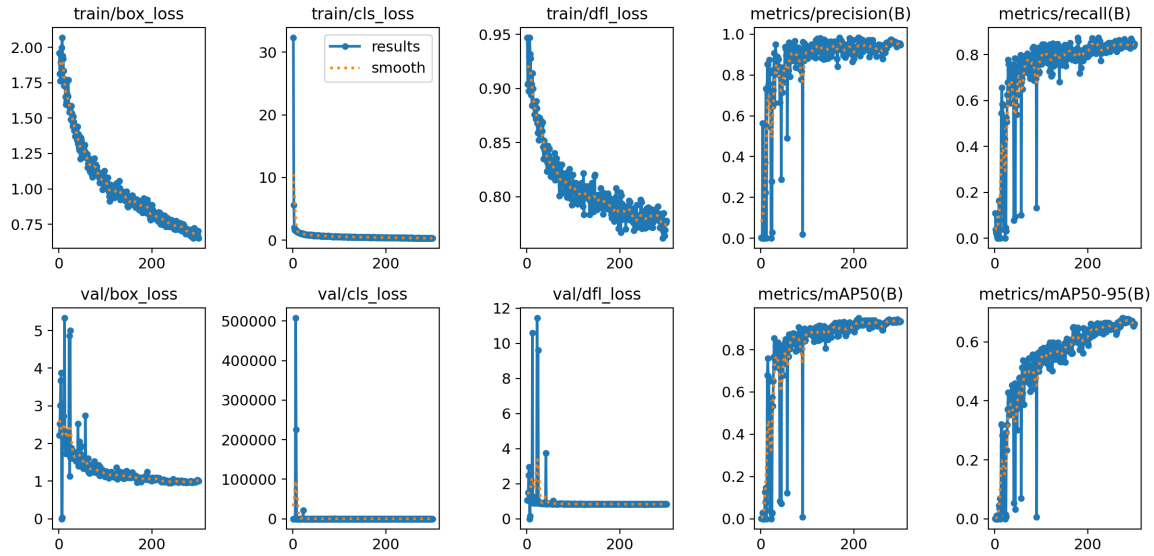

Figure 9: Metrics during training of model YOLOv11s on sliced dataset

.

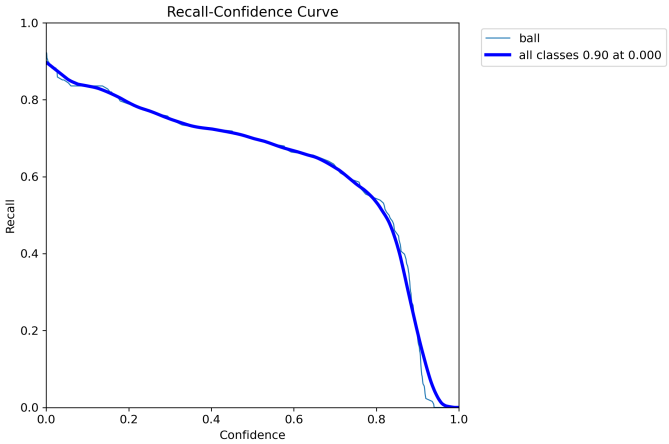

Figure 10: Recall-Confidence curve for model YOLOv11s trained on sliced dataset

.

Figure 11: Detection on validation data using model YOLOv11s. We can see that model detects round object in background and part of head, resulting in lower precisionx

References

[1] Analyzing setting trajectories with balltime ai volleyball app - youtube.

[2] Fatih Cagatay Akyon, Cemil Cengiz, Sinan Onur Altinuc, Devrim Cavusoglu, Kadir Sahin, and Ogulcan Eryuksel. SAHI: A lightweight vision library for performing large scale object detection and instance segmentation, November 2021.

[3] Glenn Jocher and Jing Qiu. Ultralytics yolo11, 2024.

[4] Maxim Tkachenko, Mikhail Malyuk, Andrey Holmanyuk, and Nikolai Liubimov. Label Studio: Data labeling software, 2020-2025. Open source software available from https://github.com/HumanSignal/label-studio.