Next-Generation Object Detection: a Review of Performance and Advanced Models Against Standards in Smart Agriculture

Paula Cecilia Martínez[1], Marcos Montoya[2,3], Julieta Dalmasso[2], and Emmanuel N. Millán[1,4]♠

[1] Instituto Tecnológico Universitario (ITU), Universidad Nacional de Cuyo, Mendoza, Argentina
[2] INTA, Mendoza, Argentina
[3] Facultad de Ciencias Agrarias, Universidad Nacional de Cuyo, Mendoza, Argentina
[4] Instituto Interdisciplinario de Ciencias Básicas ICB, CONICET-FCEN UNCuyo, Padre Contreras 1300, Mendoza 5500, Argentina
emmanuel.millan@itu.uncu.edu.ar

Abstract. Precision agriculture incorporates computer vision techniques to automate image analysis and optimize monitoring tasks in production systems. This paper presents a comparison of object detection models applied to a practical precision agriculture case study, using a proprietary dataset of grape bunch images, with a methodology extensible to standard datasets such as COCO. Four representative approaches were evaluated: Faster R-CNN as a classic two-stage model, YOLOv11 as a widely used single-stage detector, and two more recent transformer-based architectures, DINO and D-FINE. The comparison was performed considering accuracy metrics, such as mAP@50, and computational performance metrics, including inference speed. The results show that YOLOv11n offers the best compromise between accuracy and efficiency, outperforming the other evaluated models. Faster R-CNN exhibits lower performance, comparable to that of DINO, while D-FINE achieves higher accuracy at the cost of significantly higher computational resources. These results highlight the importance of selecting the detection model based on the application context and the operational constraints specific to the agricultural environment.

Keywords: Deep Learning · Object detection · Grape bunches

Detección de objetos de última generación: una revisión del rendimiento y modelos avanzados frente a los estándares en la agricultura inteligente

Abstract. La agricultura de precisión incorpora técnicas de visión artificial para automatizar el análisis de imágenes y optimizar las tareas de monitorización en los sistemas de producción. Este artículo presenta una comparación de modelos de detección de objetos aplicados a un caso práctico de agricultura de precisión, utilizando un conjunto de datos propio de imágenes de racimos de uva, con una metodología extensible a conjuntos de datos estándar como COCO. Se evaluaron cuatro enfoques representativos: Faster R-CNN como un modelo clásico de dos etapas, YOLOv11 como un detector de una sola etapa ampliamente utilizado, y dos arquitecturas más recientes basadas en transformadores, DINO y D-FINE. La comparación se realizó considerando métricas de precisión, como mAP@50, y métricas de rendimiento computacional, incluyendo la velocidad de inferencia. Los resultados muestran que YOLOv11n ofrece el mejor equilibrio entre precisión y eficiencia, superando a los demás modelos evaluados. Faster R-CNN presenta un rendimiento inferior, comparable al de DINO, mientras que D-FINE logra una mayor precisión a costa de un consumo de recursos computacionales significativamente mayor. Estos resultados resaltan la importancia de seleccionar el modelo de detección en función del contexto de la aplicación y las restricciones operativas específicas del entorno agrícola.

Keywords: Aprendizaje Profundo · detección de objetos · racimos de uvas

1 Introduction

Precision agriculture has emerged as a response to the need to optimize resource use, improve yields, and reduce the environmental impact of traditional farming practices. Through the integration of digital technologies (sensors, satellite imagery, drones, and geographic information systems), this approach allows for data-driven decision-making based on concrete and up-to-date information, adapted to the spatial and temporal variability of crops. In this context, image analysis plays a central role [15,26], enabling tasks such as monitoring crop health, detecting weeds, identifying diseases, and estimating yields.

Within this general framework, this work focuses on a specific practical case of object detection in images applied to precision agriculture, using a dataset created by the authors of this work and available online at [21]. However, the proposed evaluation, comparison, and analysis methodology is general and can be applied without substantial modifications to standard datasets widely used in the literature, such as the COCO dataset, allowing the results obtained to be placed in a broader and more comparable context.

Object detection in images is one of the classic tasks of computer vision and consists of locating and classifying object instances within an image [32]. Early approaches relied on manual feature extraction methods, followed by traditional

classifiers. With the consolidation of deep learning, models based on convolutional neural networks emerged, marking a turning point in the field. Among these, two-stage architectures, such as R-CNN and its successors, introduced the concept of generating region proposals and then classifying them, achieving significant improvements in accuracy.

Faster R-CNN represents one of the last and most influential models in this "classic" family [24]. By integrating proposal generation within the network itself, it became a benchmark for several years in both research and practical applications. Its robustness and accuracy led to its widespread adoption in various domains, including Smart Farming applications such as fruit detection, plant counting, and aerial image analysis.

Subsequently, single-stage approaches emerged, among which YOLO (You Only Look Once) became the most widely used model in practice [23]. Its main contribution was to reframe the detection problem as a single, direct prediction on the entire image, prioritizing simplicity and computational efficiency. Thanks to its balance between speed and accuracy, YOLO became a widely used tool in both academic and industrial settings, particularly in agricultural applications where processing large volumes of images is a key factor.

More recently, the introduction of transformer-based architectures opened a new line of research in object detection. These models, originally inspired by natural language processing, allow for the modeling of global relationships within an image without relying exclusively on local convolutional operations. Within this family, approaches derived from DETR (DEtection TRansformer) propose a different formulation of the problem, eliminating traditional components such as region proposals and post-processing based on non-maximum suppression. This paper considers two representatives of this approach: DINO, a robust and stable evolution of the DETR approach [35], and D-FINE, a more recent model that introduces improvements focused on accuracy and refinement of predictions [22].

In this regard, this paper presents a brief comparative review of different object detection approaches: a classic, widely validated model such as Faster R-CNN (Detectron2 implementation), a currently widely used model such as YOLO (Ultralytics), and two more recent transformer-based proposals, DINO and D-FINE. Through this comparison, we aim to analyze their relative performance in a precision agriculture scenario, providing a practical overview of the advantages and limitations of each approach.

This work first describes the pipeline used for image preparation, training, and inference in the section 2.1. The datasets used are then described, and a description of the models is provided in the section 2.3. The section 2.4 describes the hardware and software used. The results are presented in the section 3, divided into four parts: first, the time and VRAM consumption values during training; then, the quality and accuracy metrics from the testing stage; the computational performance during the inference stage; and finally, a comparison between the five models regarding their computational performance during detection and the quality of the detections (FPS vs. mAP@50) in the section 3.4. Finally, the section 4 contains the conclusions and outlines future work.

2 Materials and Methods

This section describes the pipeline used to perform the analysis in this study, including the dataset, image augmentations, hardware and software infrastructure, and metrics. Finally, it describes the five detection models used: Faster R-CNN, YOLOv11n, YOLOv11x, DINO, and D-FINE.

2.1 Pipeline

The pipeline for this study consists of 6 stages and is summarized in Figure 1:

1. Augmentation: Augmentations are used to generate a dataset with a seed that applies filters to the images.
2. YOLO to COCO: The labels are converted from YOLO format to COCO format so they can be used in the five models.
3. Validation: The dataset is verified, ensuring that the images are not corrupted and that the labels for each image have correct coordinate values.
4. Training: Training is performed by applying augmentations to the images with a seed so that all five models receive the same dataset.
5. Evaluation: The test partition of the dataset is used to evaluate the performance of the five models.
6. Unifying Metrics: A script is used to unify the results of the evaluation stage for all five models into a single file.

This pipeline can be run N times with different seeds to perform training with variation and make inferences on the test dataset, then average the resulting metrics.

2.2 Dataset

The dataset used consists of smartphone images captured in both shade and full sun. The images were taken at a distance of between 40 and 60 cm from the trellis system. They are 4160x3120 pixels and were then resized to 640 pixels wide while maintaining the aspect ratio. The images are of red grape bunches, and some include hail netting. The dataset is available at [21] and comprises 700 images: 500 for training, 100 for validation, and 100 for testing.

The images were labeled using LabelImg [29] software in YOLO format and then converted to COCO format for use with the five detection models (described in 2.3 section). Within the pipeline, image augmentation can be performed using various filters. In this work, the following filters were used: horizontal flip, vertical flip, brightness and contrast, blur, noise, and CLAHE (Contrast Limited Adaptive Histogram Equalization). These filters are applied using a seed that allows for the creation of different datasets (by varying the filters) and also allows for the generation of the same datasets for reproducibility of the results.

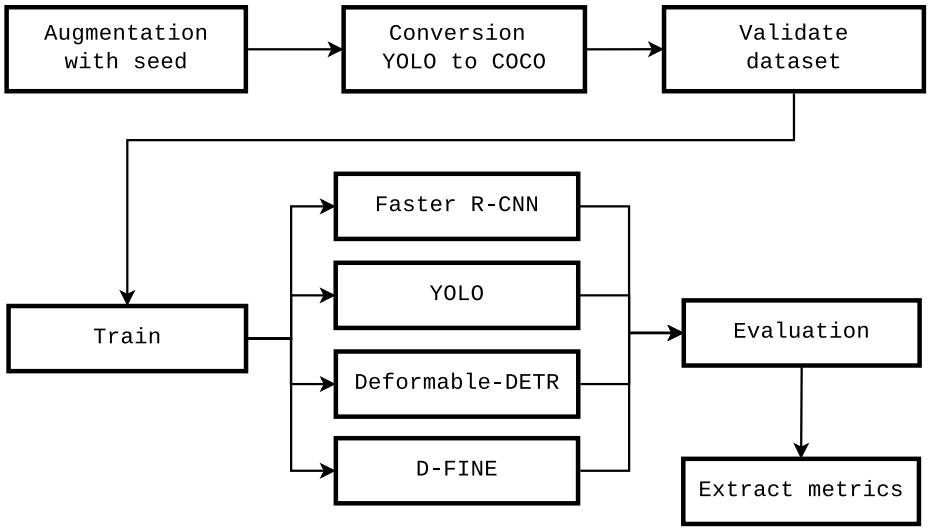

Fig. 1. Pipeline stages starting with validation and checksum generation for the dataset, to training and evaluation, and finally metrics extraction

2.3 Models

The five models analyzed in this work were: Faster R-CNN, YOLO (v11n and v11x), D-FINE, and DINO. Default configuration parameters were used for all five models, and training was performed for 300 epochs. To obtain statistics with different augmented datasets, five training runs were conducted using different seeds.

Faster R-CNN represents a significant advancement in the field of object detection, building upon the established R-CNN and Fast R-CNN models to create a more efficient and accurate deep learning architecture. This model introduces the Region Proposal Network (RPN), which enables simultaneous training of the proposal generation and object detection stages. This results in significant improvements in detection times and accuracy compared to its predecessors.

The RPN shares convolutional characteristics with the classification network, enabling it to produce high-quality region proposals directly from feature maps, eliminating reliance on external proposal generation methods such as selective search [12,6,27]. This efficient design is the essence of Faster R-CNN, integrating proposal and detection processes into a single, cohesive deep learning framework.

The Faster R-CNN architecture has been crucial in various applications, from pedestrian detection to vehicle detection in aerial imagery [34,17]. Its design allows for an end-to-end training approach, enabling the network to simultaneously learn detection and proposal generation tasks, thus reducing the computational load associated with using separate networks [17,6]. This work uses the Detectron2 implementation available at [33].

YOLO (You Only Look Once) represents a significant paradigm shift in object detection. Its architecture is unique because it addresses detection as a single regression problem, rather than treating it as a traditional classification task.

The original approach, presented by Joseph Redmon and colleagues in 2016, framed object detection in terms of spatially separated bounding boxes and their associated class probabilities, enabling real-time performance suitable for various applications, such as autonomous driving and live surveillance [23,28]. YOLO's efficiency stems from its single-stage model, which processes an entire image in a single evaluation, drastically reducing computation time compared to its multi-stage competitors, such as Faster R-CNN [9]).

Over the years, YOLO has undergone several improvements, with key iterations such as YOLOv2 and YOLOv3 introducing notable advancements in speed and accuracy. YOLOv4, released in 2020, integrated advancements in training methodologies and network architecture, further pushing the boundaries of detection performance [9]. Each version has been characterized by fine-tuning performance in benchmarks such as MS COCO and PASCAL VOC, while maintaining the high inference speeds required for real-time applications [9,28]. More recently, Ultralytics developed YOLOv5, which garnered significant attention for its performance and versatility across various platforms [16]. This series culminated with the release of YOLOv8 in 2023, a version that builds on previous versions, with improved accuracy and speed metrics, consolidating its real-time capabilities and broad application spectrum [11,25].

Recent studies have favorably compared YOLOv8 with its predecessors and other cutting-edge models, demonstrating its excellence in various fields, such as forest fire detection [5], robotic applications [36] and traffic analysis [13]. YOLOv11 is the latest evolution in the family, incorporating advancements to improve both accuracy and computational efficiency. YOLOv11 was optimized for feature extraction, achieving improvements in speed and detection capabilities compared to previous versions, particularly YOLOv8, v9, and v10 [18,4].

With its innovative design, YOLOv11 is compatible with a wide variety of uses, including real-time monitoring in agriculture. For example, recent studies have shown that it can detect apple scab under various lighting conditions [18]. Furthermore, its broad range of applications extends to medical imaging, where it assists in the diagnosis of conditions such as coronary artery disease by accurately segmenting vessel characteristics in angiograms [10].

YOLOv11 also places great importance on user accessibility, offering a simplified training process that allows users to fine-tune pre-trained models on their own datasets very effectively [4,7]. This flexible architecture contributes significantly to YOLOv11's robustness in practical scenarios, ensuring that it meets the demands of real-time object detection in various fields.

The YOLO framework remains a fundamental architecture in computer vision today, continually influencing new research and real-world applications.

DINO (Deformable DETR) is an extension of the original DETR (Detection Transformer) architecture that enhances its object detection capabilities through

a deformable attention mechanism. This technique allows the model to adaptively sample features in different spatial locations, which is essential for effectively handling diverse object shapes and visual contexts.

Introduced by Zhu and colleagues in 2020, Deformable DETR significantly reduced the complexity of the initial DETR model. This is because it improved its ability to handle scenarios with many objects together without requiring excessive computational resources [37]. This approach proved to accelerate the convergence rate during training, allowing the model to achieve high accuracy in fewer epochs compared to traditional DETR implementations [19].

It is worth noting that DINO has gained recognition for its uses in diverse fields such as microfluidics, where it is successfully used to detect and track microdroplets. It also adapts to complex environments to monitor various physiological conditions [31]. Furthermore, adaptations of Deformable DETR have been implemented in drone (UAV) scenarios for obstacle detection, demonstrating its versatility in different application areas and achieving notable improvements in object detection performance [20,30].

D-FINE is a framework ("Redefine regression task in DETRs as fine-grained distribution refinement") that represents a significant advancement in real-time object detection using DETR-type models [22]. Its main focus is on correcting a historical weakness of these architectures: object localization (boundary box regression). To achieve this, D-FINE relies on two innovations that work together to improve accuracy and optimization during training.

The first is Fine-Grained Distribution Refinement (FDR), which abandons the idea of predicting fixed coordinates. Instead, it transforms the regression task into an iterative process that refines probability distributions for each edge of the box (like fine-tuning, layer by layer). This allows the model to become progressively more accurate. The second innovation is Global Optimal Location Self-Distillation (GO-LSD). This is a self-distillation mechanism that transfers location knowledge from the deepest, most accurate layers (the "masters") to the surface layers (the "learners"). This ensures the model starts with better predictions from the outset, accelerating training and improving final accuracy.

In practice, D-FINE is delivering excellent results for real-time detection. For example, the D-FINE-X variant achieves an impressive 55.8% AP (average accuracy) at 78 frames per second (FPS) in the COCO benchmark [22], outperforming other real-time detectors. This makes it a very powerful and versatile tool. As a high-performance, general-purpose detector, it has direct applications in areas ranging from surveillance and robotics (such as autonomous vehicles) to industrial inspection [8], where extremely fast and highly accurate detection is required. Because it is a recent development (late 2024), there are few published applications for this model.

2.4 Software and hardware

The hardware used for this study consists of a workstation with an AMD Ryzen 7 2700 processor, 64GB of DDR4 3200MHz RAM, and a Gigabyte GeForce RTX

4080 SUPER GAMING OC 16GB GPU (with the factory default settings, no overclocking). The operating system installed is Linux Slackware 15 with Python 3.11.10 and PyTorch 2.9.1+cu128.

The software Ultralytics 8.3.234 [14] was used to run the YOLO v11n (nano) and v11x (extra-large) models. Detectron2 version 0.6, an implementation of the Faster R-CNN model available at [1] and developed by Facebook, was used. The DINO implementation used is provided by the transformers package version 4.57.3, and the pre-trained model "SenseTime/deformable-detr" was used [37,2]. Finally, the official D-FINE implementation available on GitHub [22,3] branch master commit d669475 was used, model version hgnetv2_n. Larger model versions did not fit in the GPU's VRAM (16 GB). All models (except DINO) were run with a batch size of 8. DINO was run with a batch size of 1; larger batch sizes could not be executed on the GPU used in this work (the model ran out of VRAM).

3 Results and Discussion

3.1 Training

The training process for the five models was performed on a single GPU (see 2.4). Five training runs were conducted, varying the seed for the image augmentation process, and the training for all five models lasted 300 epochs. In the case of Ultralytics YOLO, the 'patience' variable was set to 0 to prevent training from stopping before reaching 300 epochs and to ensure the same number of epochs were run as for the other models. Ultralytics stops training if it detects (by default) that there is no improvement in the training during the last 100 epochs; changing the "patience" variable to 0 disables this behavior.

Table 1 shows the average training time and VRAM consumption in GB for each model. A clear difference in training time between the models can be observed. YOLOv11n is the fastest of the five models, $\sim$ 13 times faster than DINO, $\sim$ 6 times faster than Detectron2, and $\sim$ 3 times faster than D-FINE. YOLOv11n also has the lowest memory consumption, using less than 2GB of VRAM, half that of DINO and $\sim$ 3.5 times less than Detectron2. When comparing YOLO v11n to v11x, the nano version (v11n) is $\sim$ 2.5 times faster than v11x and uses $\sim$ 4.7 times less VRAM.

Table 1. Wallclock time (seconds) and maximum VRAM (GB) used during training for each model, average for 5 training datasets.

Model	Time (s)	Max VRAM (GB)
Detectron2	9993 $\pm$ 20	7.1 $\pm$ 0.5
YOLO v11n	1700 $\pm$ 26	1.97 $\pm$ 0.02
YOLO v11x	4299 $\pm$ 26	9.3 $\pm$ 0.01
DINO	22079 $\pm$ 173	4.0 $\pm$ 0
D-FINE	5030 $\pm$ 24	3.6 $\pm$ 0.07

In the following sections, the accuracy and detection quality metrics of each model will be analyzed to assess whether this difference in performance during training translates into lower detection quality.

3.2 Detection Accuracy and Quality Metrics

To compare the results between the models, the following detection performance metrics were used: precision, mAP@50, and mAP@50-95. The first analysis focused on the quality of the detections made during the inference stage on the test dataset (images that the model never saw during the training stage). The results for three metrics (precision, mAP@50, and mAP@50-95) and the five models are summarized in Figure 2. It is noteworthy that the YOLOv11 model outperforms the other models in all three metrics, achieving a precision of over 0.7 (for v11x). YOLOv11n has similar values to v11x. D-FINE follows, and then Detectron2 and DINO, with very similar results for all three metrics. Given the number of images used to train the models, acceptable precision and mAP@50 were expected (precision in the range of 0.5 to 0.6 and mAP@50 greater than 0.4). In the case of D-FINE and YOLO, both exceed 0.6 for mAP@50. Increasing the number and variety of images in the training dataset is necessary to improve these results.

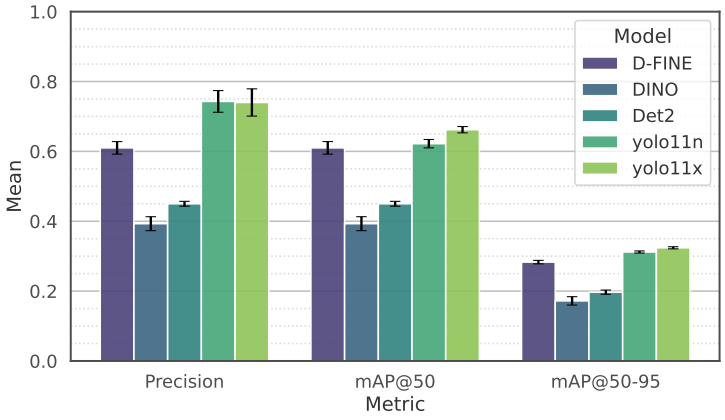

Fig. 2. Average of 5 detections with the precision metrics, mAP@50 and mAP@50-95 for the five models.

3.3 Computational Performance

Regarding the computational performance of the five models, the VRAM consumption in GB can be seen in Figure 3. Detectron2 used the least memory

during the inference stage ($\sim$ 0.4 GB), followed by YOLOv11n ($\sim$ 0.55 GB), then D-FINE ($\sim$ 0.75 GB), DINO reached almost 1 GB of VRAM consumption, and finally YOLOv11x exceeded 2.5 GB of VRAM.

Regarding detection speed, YOLOv11n is capable of processing $\sim 450 \pm 19$ FPS (frames per second), YOLOv11x reaches $\sim 91 \pm 1$ FPS, D-FINE can detect $\sim 11 \pm 0.1$ FPS, DINO $\sim 14 \pm 0.1$ FPS and finally Detectron2 can detect $\sim 37 \pm 0.2$ FPS.

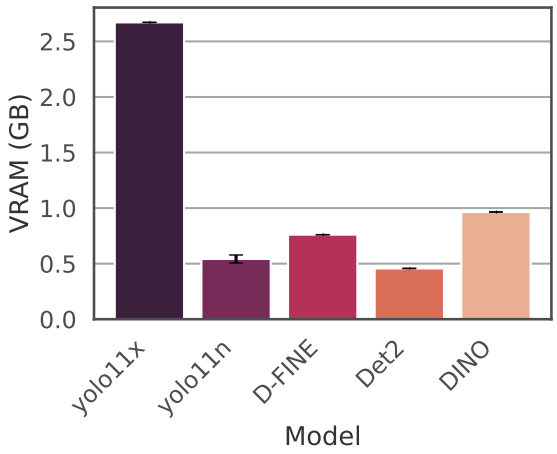

Fig. 3. Average GPU RAM usage used in inference for the five models.

When analyzing VRAM usage and detections per second, YOLOv11n is considerably more efficient than the other models, only slightly surpassed by detectron2 with lower memory consumption. Regarding FPS, YOLO (in both versions) is between 5 and 40 times faster than the other three models. The next section compares the detection accuracy at mAP@50 for each model and its FPS performance.

3.4 Accuracy-Efficiency Trade-off Analysis

A quick and easy way to compare detection performance and accuracy can be seen in Figure 4. The x-axis shows the number of FPS each model can process during inference; higher values are better. The y-axis shows accuracy measured at mAP@50; higher values are better. The DINO and Detectron2 models have very similar mAP@50 values and very close inference performance. D-FINE has higher accuracy (close to 0.6) and a similar FPS level to the other two models. YOLO v11n has a high mAP@50 value of $\sim$0.65 and a detection performance of $\sim$450 FPS. Finally, YOLO v11x has an mAP@50 value of 0.66 (the highest

of all the models) but its performance drops to ∼92 FPS. YOLO's FPS values allow for real-time detection.

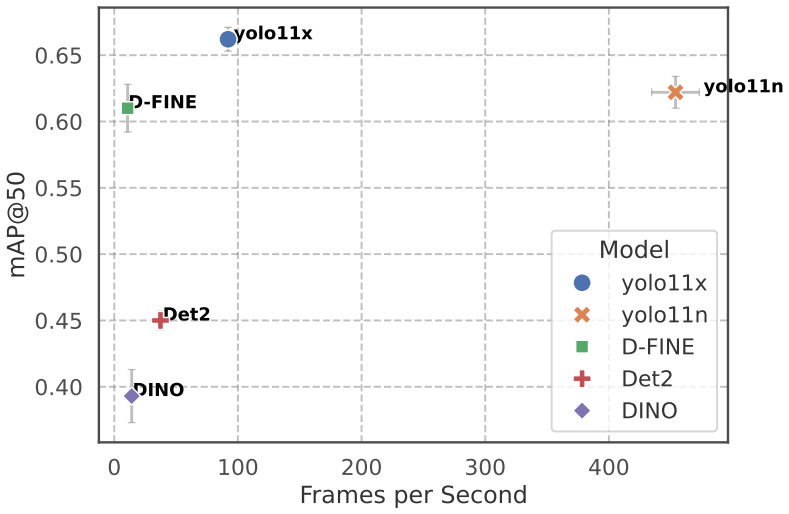

Fig. 4. Comparison of inference performance and the accuracy obtained for each model.

Table 2. Values of metrics used in the work, average inference over the five trainings for the five models.

Model	mAP@50	mAP@50-95	Precision	FPS	VRAM (GB)
yolo11n	0.62 ± 0.01	0.31 ± 0.003	0.74 ± 0.03	453 ± 19	0.54 ± 0.036
yolo11x	0.66 ± 0.009	0.32 ± 0.003	0.74 ± 0.03	92 ± 0.5	2.67 ± 0
D-FINE	0.61 ± 0.01	0.28 ± 0.005	0.61 ± 0.01	10 ± 0.08	0.76 ± 0
Detectron2	0.45 ± 0.007	0.19 ± 0.006	0.45 ± 0.007	37 ± 0.3	0.45 ± 0
DINO	0.39 ± 0.02	0.17 ± 0.01	0.39 ± 0.02	14 ± 0.02	0.96 ± 0

Tests performed with the grape cluster dataset show that YOLOv11n offers higher computational performance and detection accuracy than the other three models. The classic Faster R-CNN (detectron2 implementation) has low performance (both in FPS and mAP@50) compared to YOLO and similar performance to DINO. D-FINE significantly improves accuracy but is up to 40 times slower than YOLO in detection.

Given these results, and considering its ease of use and accessibility, YOLO remains the recommended choice. Regarding model licensing, Ultralytics uses the AGPL-3.0 license, which generally requires that any modifications to the code

and weights must also be open source and available. To use the models provided by Ultralytics commercially, an Enterprise license must be purchased (`https://www.ultralytics.com/es/license`). On the other hand, DETR models use the Apache 2.0 license (`https://github.com/facebookresearch/detr`), which is more permissive than AGPL-3.0; it is not necessary to distribute the modified model code or weights. This difference in license can be a deciding factor if computational performance is not the primary metric when selecting a model.

4 Conclusions and Future Work

In line with the main objective of this work, which is to compare different object detection approaches applied to a specific precision agriculture case, the results obtained on the grape bunch dataset allow us to draw clear conclusions regarding the trade-off between accuracy and computational performance. Based on the tests performed, YOLOv11n exhibits the best overall performance, simultaneously achieving the highest detection accuracy and the greatest inference speed compared to the other models evaluated. Meanwhile, the classic Faster R-CNN model, implemented using Detectron2, shows inferior performance in both FPS and mAP@50, with values clearly below YOLO and comparable to those obtained by DINO. This reflects the limitations of two-stage architectures in scenarios where efficiency is a relevant factor. Finally, D-FINE stands out for its considerable improvement in detection accuracy, although at the cost of significantly higher computational resources, becoming up to forty times slower than YOLO during inference. These results reinforce the idea that the choice of model depends heavily on the application context and possibly the model's licensing, and that in practical Smart Farming scenarios, where efficient processing of large volumes of images is prioritized, models like YOLO continue to offer an accuracy-performance ratio that is hard to match.

This work can be complemented with optimizations to the D-FINE model, such as training it in PyTorch as done here, but exporting it to the ONNX format and then compiling it with TensorRT. We will continue exploring YOLOv11 and future versions in real-time environments, acquiring drone images and performing inference on on-premises hardware such as Raspberry Pi 5 with Hailo AI and Arduino Uno Q. This would allow us to analyze field images and obtain detection results in near real-time, without needing to access cloud services, which is especially relevant in rural areas where internet connections are often low-bandwidth or nonexistent.

Acknowledgments. This study was supported by Universidad Nacional de Cuyo projects number SIIP T001-T4 2022/2024 and SIIP 80020240400062UN. This work used the Toko Cluster from FCEN, UNCuyo, which is part of the SNCAD, MinCyT, Argentina. TOKO cluster was expanded in CPU and GPU capacity using funds from the REMATE project (CONVE-2023-100765538-APN-MCT) led by Dr. Pablo Mininni (IP-Nodo Bs As) and Dr. Rafael P. Fernandez (IP-Nodo Mendoza).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

CRediT author statement

Paula Cecilia Martínez: Conceptualization, Investigation, Methodology, Software, Visualization, Writing – original draft preparation. Marcos Montoya: Data curation, Funding acquisition, Investigation, Methodology, Validation, Writing – review and editing. Julieta Dalmasso: Data curation, Investigation, Methodology, Validation, Writing – review and editing. Emmanuel N. Millán: Conceptualization, Data curation, Funding acquisition, Investigation, Methodology, Software, Supervision, Visualization, Writing – original draft preparation, Writing – review and editing.

References

1. Detectron2 facebook ai research (2021), https://github.com/facebookresearch/detectron2
2. Hugging face transformers deformable-detr (2022), https://huggingface.co/docs/transformers/v4.56.0/model_doc/deformable_detr
3. D-fine: Redefine regression task of detrs as fine-grained distribution refinement (2024), https://github.com/Peterande/D-FINE
4. Ali, M.L., Zhang, Z.: The yolo framework: A comprehensive review of evolution, applications, and benchmarks in object detection. Computers **13**(12), 336 (2024)
5. Casas, E., Ramos, L., Bendek, E., Rivas, F.: Assessing the effectiveness of yolo architectures for smoke and wildfire detection. Ieee Access (2023). https://doi.org/10.1109/access.2023.3312217
6. Chang, Y.L., Amare, A., Chang, L., Wang, Y.C., Hsiao, C.Y., Lee, W.H.: Ship detection based on yolov2 for sar imagery. Remote Sensing (2019). https://doi.org/10.3390/rs11070786
7. Dalmasso, J., Montoya, M.A., Martínez, P.C., Millán, E.N.: Anthill detection in precision agriculture using machine learning. In: 2025 XXI Workshop on Information Procesing and Control (RPIC). pp. 1–5 (2025). https://doi.org/10.1109/RPIC67987.2025.11260763
8. Dang, L.M., Sagar, A.S., Bui, N.D., Nguyen, L.V., Nguyen, T.H.: Attention-guided marine debris detection with an enhanced transformer framework using drone imagery. Process Safety and Environmental Protection **197**, 107089 (2025)
9. Diwan, T., Anirudh, G., Tembhurne, J.V.: Object detection using yolo: Challenges, architectural successors, datasets and applications. Multimedia Tools and Applications (2022). https://doi.org/10.1007/s11042-022-13644-y
10. Díaz-Gaxiola, E., Yee-Rendón, A., Vega-López, I.F., Campos-Leal, J.A., García-Aguilar, I., López-Rubio, E., Luque-Baena, R.M.: Experimental assessment of yolo variants for coronary artery disease segmentation from angiograms. Electronics (2025). https://doi.org/10.3390/electronics14132683
11. Gupta, A., Mhala, P., Mangal, M., Yadav, K., Sharma, S.: Traffic sign sensing: A deep learning approach for enhanced road safety (2024). https://doi.org/10.21203/rs.3.rs-3889986/v1
12. Hachchane, I., Badri, A., Sahel, A., Ruichek, Y.: Large-scale image-to-video face retrieval with convolutional neural network features. Iaes International Journal of Artificial Intelligence (Ij-Ai) (2020). https://doi.org/10.11591/ijai.v9.i1.pp40-45

13. He, C., Saha, P.: Investigating yolo models towards outdoor obstacle detection for visually impaired people. arXiv preprint arXiv:2312.07571 (2023). https://doi.org/10.21203/rs.3.rs-3733857/v1
14. Jocher, G., Qiu, J.: Ultralytics yolo11 (2024), https://github.com/ultralytics/ultralytics
15. Karunathilake, E.M.B.M., Le, A.T., Heo, S., Chung, Y.S., Mansoor, S.: The path to smart farming: Innovations and opportunities in precision agriculture. Agriculture (2023). https://doi.org/10.3390/agriculture13081593
16. Khachatrian, E., Sandalyuk, N., Lozou, P.: Eddy detection in the marginal ice zone with sentinel-1 data using yolov5. Remote Sensing (2023). https://doi.org/10.3390/rs15092244
17. Khan, S.A., Lee, H.J., Lim, H.: Enhancing object detection in self-driving cars using a hybrid approach. Electronics (2023). https://doi.org/10.3390/electronics12132768
18. Leiva, F., Rebeaud, S.G., Christen, D.: Real-time identification and quantification of apple scab on fruit in preharvest and postharvest conditions using yolov11: A deep learning approach (2025). https://doi.org/10.21203/rs.3.rs-7010152/v1
19. Li, F., Zhang, H., Liu, S., Guo, J., Ni, L.M., Zhang, L.: Dn-detr: Accelerate detr training by introducing query denoising. Ieee Transactions on Pattern Analysis and Machine Intelligence (2024). https://doi.org/10.1109/tpami.2023.3335410
20. Liu, X., Li, H.: A study on uav target detection and 3d positioning methods based on the improved deformable detr model and multi-view geometry. Advances in Mechanical Engineering (2025). https://doi.org/10.1177/16878132251315505
21. Millán, E.N., Montoya, M.A., Martinez, P.C., Dalmasso, J., Parlanti, T.S.: Imagenes de racimos de uva etiquetadas con formato YOLO (2025). https://doi.org/http://hdl.handle.net/11336/254683
22. Peng, Y., Li, H., Wu, P., Zhang, Y., Sun, X., Wu, F.: D-fine: Redefine regression task in detrs as fine-grained distribution refinement. arXiv preprint arXiv:2410.13842 (2024)
23. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection (2016). https://doi.org/10.1109/cvpr.2016.91
24. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems **28** (2015)
25. Sharma, N., Baral, S., Paing, M.P., Chawuthai, R.: Parking time violation tracking using yolov8 and deepsort (2023). https://doi.org/10.20944/preprints202305.0828.v1
26. Szira, Z., Varga, E., Csegődi, T.L., Milics, G.: The benefits, challenges and legal regulation of precision farming in the european union. Eu Agrarian Law (2023). https://doi.org/10.2478/eual-2023-0001
27. Tang, T., Zhou, S., Deng, Z., Zou, H., Lei, L.: Vehicle detection in aerial images based on region convolutional neural networks and hard negative example mining. Sensors (2017). https://doi.org/10.3390/s17020336
28. Terven, J., Córdova-Esparza, D., Romero-González, J.A.: A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. Machine Learning and Knowledge Extraction (2023). https://doi.org/10.3390/make5040083
29. Tzutalin: Labelimg: A graphical image annotation tool to label object bounding boxes in images (2015), https://github.com/tzutalin/labelImg

30. Wang, D., Li, Z., Du, X., Ma, Z., Liu, X.: Farmland obstacle detection from the perspective of uavs based on non-local deformable detr. Agriculture (2022). https://doi.org/10.3390/agriculture12121983
31. Wang, J., Wang, H., Lai, H., Liu, F.X., Cui, B., Yu, W., Mao, Y., Yang, M., Yao, S.: A machine vision perspective on droplet-based microfluidics. Advanced Science (2025). https://doi.org/10.1002/advs.202413146
32. Wu, X., Sahoo, D., Hoi, S.C.H.: Recent advances in deep learning for object detection. Neurocomputing **396**, 39–64 (2020). https://doi.org/10.1016/j.neucom.2020.01.085
33. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019)
34. Yan, H., Chen, C., Jin, G., Zhang, J., Wang, X., Zhu, D.: Implementation of a modified faster r-cnn for target detection technology of coastal defense radar. Remote Sensing (2021). https://doi.org/10.3390/rs13091703
35. Zhang, H., Li, F., Liu, S., Zhang, L., Su, H., Zhu, J., Ni, L.M., Shum, H.Y.: Dino: Detr with improved denoising anchor boxes for end-to-end object detection (2022)
36. Zhou, Q., Liu, D., An, K.: Ese-yolov8: A novel object detection algorithm for safety belt detection during working at heights. Entropy (2024). https://doi.org/10.3390/e26070591
37. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020)