agriculture

MDPI

Article

Towards Intelligent Pruning of Vineyards by Direct Detection of Cutting Areas

Elia Pacioni [1,2], Eugenio Abengózar [3], Miguel Macías Macías [1,4,*], Carlos J. García-Orellana [3,4], Ramón Gallardo [4,5] and Horacio M. González Velasco [4,5]

1 Centro Universitario de Mérida, Universidad de Extremadura, Avda. Santa Teresa de Jornet, 38, 06800 Mérida, Spain; eliapacioni@unex.es
2 HES-SO Valais/Wallis, 3960 Sierre, Switzerland
3 Facultad de Ciencias, Universidad de Extremadura, Avda. de Elvas, s/n, 06006 Badajoz, Spain; eabengozar@unex.es (E.A.); cjgarcia@unex.es (C.J.G.-O.)
4 Instituto de Computación Científica Avanzada, Av. de la Investigación, s/n, 06006 Badajoz, Spain; rgallardo@unex.es (R.G.); hmgvelas@unex.es (H.M.G.V.)
5 Escuela Politécnica, Universidad de Extremadura, Avda. Universidad s/n, 10003 Cáceres, Spain
* Correspondence: mmacias@unex.es

Abstract: The development of robots for automatic pruning of vineyards using deep learning techniques seems feasible in the medium term. In this context, it is essential to propose and study solutions that can be deployed on portable hardware, with artificial intelligence capabilities but reduced computing power. In this paper, we propose a novel approach to vineyard pruning by direct detection of cutting areas in real time by comparing Mask R-CNN and YOLOv8 performances. The studied object segmentation architectures are able to segment the image by locating the trunk, and pruned and not pruned vine shoots. Our study analyzes the performance of both frameworks in terms of segmentation efficiency and inference times on a Jetson AGX Orin GPU. To compare segmentation efficiency, we used the mAP50 and AP50 per category metrics. Our results show that YOLOv8 is superior both in segmentation efficiency and inference time. Specifically, YOLOv8-S exhibits the best tradeoff between efficiency and inference time, showing an mAP50 of 0.883 and an AP50 of 0.748 for the shoot class, with an inference time of around 55 ms on a Jetson AGX Orin.

Keywords: computer vision; object segmentation; Mask R-CNN; YOLOv8; vineyard pruning

Academic Editors: Xiuguo Zou, Xiaochen Zhu, Wentian Zhang, Yan Qian and Yuhua Li

Received: 30 April 2025
Revised: 20 May 2025
Accepted: 23 May 2025
Published: 27 May 2025

Citation: Pacioni, E.; Abengózar, E.; Macías, M.M.; García-Orellana, C.J.; Gallardo, R.; González Velasco, H.M. Towards Intelligent Pruning of Vineyards by Direct Detection of Cutting Areas. *Agriculture* **2025**, *15*, 1154. https://doi.org/10.3390/agriculture15111154

1. Introduction

The global wine industry plays a vital role in the economy, culture, and environment. Economically, it generates millions of jobs, drives significant exports from leading producers like France, Italy, Spain, or Chile, and adds high value to global markets. The wine industry is facing several technical challenges, such as precision viticulture, climate adaptation, fermentation control, and sustainability. Also, some operational requirements in wine production such as regulatory compliance and quality assurance need to be addressed [1].

The tasks that need to be performed by skilled labor in the agricultural sector are numerous, such as phytosanitary treatments, managing crops, harvesting, and pruning. All these activities together with the shortage of expert labor create a major problem [2]. At the same time, this issue favors investment in technology (around USD 106 billion [3]) and brings the agricultural sector closer to the technological world [4].

The present study addresses the issue of automatized vineyard pruning. This section offers an analysis of the manual techniques and technological processes that have been

employed in this field to date. Subsequently, the technological advances achieved in pruning vineyards are presented, including the utilization of artificial intelligence (AI) in agricultural practices.

1.1. Vineyard Pruning

There are numerous reasons why a vineyard should be pruned. The most significant of these are growth control, grape quality, production improvement, disease prevention, and harvest facilitation [5]. There are multiple methods of growing vines outdoors (Royat hedge, Guyot hedge, vertical cordon, . . .). This study focuses exclusively on the complete or partial removal of shoots. In the cases where the shoots are partially pruned, it is important to maintain a specific number of buds (typically two), as vine renewal process initiates at these points.

Numerous companies are investing in technology, with the objective of compensating for the labor shortage through the implementation of intelligent automated systems [6]. The use of Real-Time Kinematic (RTK) technology has facilitated the mechanization of many stages of agricultural activities, starting with planting, harvesting, and concluding with plant care. This technology provides the farmer with a Global Positioning System (GPS) coordinate map, which is useful to automate planned activities.

Nevertheless, although these procedures have achieved a high degree of acceptance, there are still activities that are performed manually by specialized personnel, which increases the problem of labor shortages. Although progress has been made in the area of mechanical prepruning (as illustrated in Figure 1), which serves to simplify the subsequent work of manual pruning, full automation of this process requires major developments.

Figure 1. Prepruning machine of a Royat vineyard.

In order to guide a robotic arm in the execution of optimal cuts during vine pruning, efficient and real-time identification of shoots is necessary, which is set around 30 FPS to roughly match the speed of human perception [7]. Although human perception is taken as a reference for the frame threshold, 30 FPS is not necessary to process in this study because the technical time required for the robot to perform the pruning should be considered. Therefore, even a significantly lower frame-rate can be considered suitable. In similar studies, as presented by Botterill et al. [8], they achieved a speed of 2.5 FPS, which was deemed suitable for the task.

This can be achieved by deploying AI algorithms and computer vision (CV) technologies on portable hardware, which typically has less computing power than a desktop Graphics Processing Unit (GPU).

1.2. Related Works

In this paper, we deal with the problem of vineyard pruning, which has been treated from different points of view in the literature. We analyzed several approaches to the problem such as reconstruction of the vine structure [8,9], volumetric identification of shoots and buds [10], or detection of branches and buds [11–13].

Within the first category, Botterill et al. [8] presented a complete robotic platform that specializes in creating a three-dimensional model of vines using a structure that completely covers the plant and prunes it in 2 min. Gentilhomme et al. [9] showed a method for automatic extraction of the grapevine structure from two-dimensional images using a Stacked Hourglass Network, allowing to address the problem of having an unknown number of branches in the vine plant. They achieved a 95% precision and 90% recall for node prediction. In the category of volumetric identification of shoots and buds, Lanza et al. [10] evaluated a portable vision-based acquisition device which merges depth, tracking and RGB information to estimate the wood volume of the shoots, and You Only Look Once (YOLO) version 8 to detect buds, with a 0.79 F1-score.

In the group of papers dealing with the direct detection of shoots and buds, Villegas et at. [11] implemented a MobileNet architecture to segment buds in the images reporting a 0.88 F1-score in this task. Chen et al. [12] dealt with the identification of the shoot cutting points by locating the buds with two networks, and they reported an F1-score of 0.80 for the YOLOv5 model. Finally, Oliveira et al. [13] used four versions of YOLO to identify nodes in two-dimensional grapevine images, concluding that YOLOv7 was the best option, with an F1-score reaching a value of 0.86.

In some of the works cited above, controlled backgrounds were used [8–10,14] to improve performance, but this option increases the cost of a final prototype, as in [8]. It would be desirable to avoid this element in order to develop a prototype, as in the approaches of [12,13].

Our approach attempts to test whether an algorithm is able to directly detect the section of the vine shoot where the cutting tool should act, even in vineyards where partial pruning was performed, which complicates the task. We presented preliminary results in [15] with a *Mask Region-based Convolutional Neural Network* (Mask R-CNN), and now we extend the work by comparing performance with one of the most reliable state-of-the-art network models (YOLOv8).

1.3. Object Detection Algorithms

We compared the performance in our problem of two of the most common instance segmentation models: Mask R-CNN and YOLOv8. The choice is due to the fact that Mask R-CNN represents a milestone in the field of mask-level segmentation, while YOLO is among the fastest and most suitable models for working in real-time projects. The Mask R-CNN algorithm was developed by *Facebook AI Research* [16] in 2017, born as an extension of Faster R-CNN [17]. We used the Mask R-CNN implementation available in Detectron2 [18] with a ResNet50 backbone. The YOLO family of algorithms [19] has evolved in time improving the quality of its predictions while preserving high inference speeds. We used the YOLOv8 implementation developed by Ultralytics [20].

The main differences between the Mask R-CNN and YOLO algorithms are as follows:

- Mask R-CNN uses two neural networks, one for boxes and classes and the other for masks, while YOLO uses only one;

- Mask R-CNN uses the anchor concept to propose regions, while YOLOv8 uses an anchor-free approach;
- Mask R-CNN uses the RoIAlign technique for mask alignment, while YOLOv8 does not provide anything analogous;
- Mask R-CNN is designed for segmentation mask generation, while YOLOv8 supports mask-level segmentation, but has historically been used for object detection tasks.

Our final aim is to run the object detector embedded in a robot, so we must use a compact and a high-performance computing module such as the Jetson family developed by NVIDIA. In this work, we want to check if the YOLOv8 algorithms can improve our previous results in terms of object segmentation and inference times using an NVIDIA Jetson AGX Orin Developer Kit 64 Gb (NVIDIA, Santa Clara, CA, USA).

The rest of the document is structured as follows. Section 2 defines the starting hypothesis and aims of this work. Section 3 describes used materials, applied methodology, configurations, and metrics to evaluate the results. Section 4 presents the experiments performed and compares the two algorithms chosen to approach the problem. Finally, Section 5 presents the conclusions of the study and outlines the trajectory of future work.

2. Hypothesis and Aims

Our starting hypothesis is that we can train an object detector to locate the vine shoots present in an image, but unlike other approaches, we only try to localize the initial portion of the shoots, including the first buds. The localization of these regions will allow us to estimate the possible cutting points of the shoots in a post-processing phase. Previous results [15] have indicated that to improve the performance in the pruning process, we should include three categories of objects that appear in a partially pruned vine plant: shoots, pruned shoots, and trunk. As our algorithms must run on board a robot, we need to use an embedded computing board. An NVIDIA Jetson AGX Orin was selected for this experiment, offering an AI performance of 275 TOPS, 2048 NVIDIA Ampere cores, 64 Tensor Cores, and a 64-bit 12-core Arm Cortex-A78AE CPU (https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/, accessed on 20 May 2025).

The aim of this work can be divided into two subobjectives:

1. To implement the use of the YOLOv8 algorithm in order to address the segmentation problem and compare the results with those previously obtained using Mask R-CNN;
2. To test the trained models on a Jetson AGX Orin system and determine under which conditions the obtained results allow a real-time operation of a robotic pruner.

3. Materials and Methods

This section outlines the dataset employed, the algorithms configurations, and the metrics utilized for evaluation.

3.1. Dataset

In order to train and test the algorithms, we developed a labeled dataset of vineyard images [21]. The images were taken in the southwest of the province of Badajoz in Spain, where the pardina variety and the Royat hedge arrangement style are the most common. In this article, we worked with pardina vineyards pruned in the double Royat style, in which the shoots are vertically trellised on wires.

Images were taken using two different cameras: Canon EOS 2000D and Samsung SM-A715F (smartphone) (Suwon, Republic of Kore). The image collection began in 2021 and was repeated annually, though the 2023 images were not used in the current research. The prototype set and its ground truth (GT) were built using the VGG Image Annotator (VIA) software [22]. During the label creation process, the three classes *shoot*, *pruned shoot*,

and *trunk* were registered. Each image always contains at least one *trunk* object, along with objects from the other two classes. The *pruned shoot* class was added to assist the system in developing generalizable solutions. However, it is not guaranteed that all three categories are present in every image.

The addition of the *pruned shoot* class to our work is the result of a previous study using Mask R-CNN on an initial set of images. Early models tended to segment pruned shoots as shoots to be pruned. Labeling examples of this new class improved the performance in detecting shoots that actually need to be pruned. Figure 2 illustrates an example of a partially pruned vine plant labeled through the VIA software. As we can see, for the *shoot class*, only the initial portion of the shoot is considered, as previously stated.

Figure 2. Examples of images labeled with VIA software for classes *shoot* (yellow), *pruned shoot* (blue), and *trunk* (green).

Table 1 presents the dataset containing 536 images with different resolutions. They were taken from different positions and distances, looking for the robustness of the system. After the annotation process, as presented in Table 2, we obtained 5329 objects of the class *shoot*, 887 for *pruned shoot*, and 752 for *trunk*. This dataset [21] is available to the research community as detailed in the Data Availability Statement section.

Table 1. Images resolution.

Number of Images	Resolution
151	6000 × 4000
261	4624 × 2604
23	2048 × 1532
22	1600 × 1197
79	800 × 600

Table 2. Structure of the dataset.

Dataset	Shoots	Pruned Shoots	Trunk	Total	Images
Original	5329	887	752	6968	536
Balanced	5329	5029	1741	12,099	1288

As we can see in Table 1, the dataset is clearly imbalanced, maintaining a 6-to-1 ratio between the number of prototypes of the shoot and pruned shoot classes. This is because the pruned shoot class was not included at the beginning of our work. It is well known that machine learning models struggle to differentiate minority classes and tend to treat them as noise with respect to majority classes [23]. In [23,24], the authors showed different strategies to mitigate the class imbalanced data. In this work, we balanced the dataset by applying data augmentation on the images where objects of the *pruned shoot* class are in

the majority. In this way, when data augmentation generates a new image, the number of prototypes of the majority class is increased. However, it also increases, albeit to a lesser extent, those of the other classes present in the image. Figure 3 shows examples of the generated images.

Table 2 summarizes the new dataset distribution. Unfortunately, it is impossible to balance the instances of the trunk class, since each trunk instance is present once or, at most, twice in each image, making it impossible to augment its instances without further augmenting the other classes.

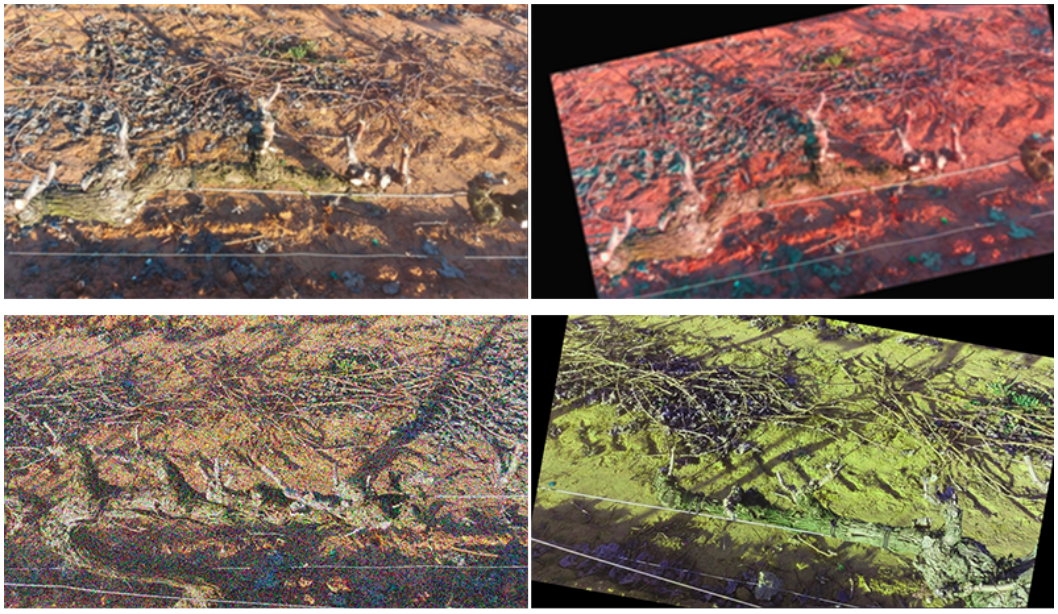


Figure 3. Example of the application of data augmentation to balance the dataset.

3.2. Algorithm Configurations

In [15], we already studied the performance of a Mask R-CNN model on a preliminary dataset, and we obtained the optimized configuration parameters that we use in this work. We trained our Mask R-CNN-FPN networks by applying the transfer learning technique, using a ResNet50 backbone provided by the Detectron2 project as baseline model id 137849600. We trained our networks using data augmentation, enabling the algorithms specified in Table 3. The optimized *anchor sizes* were 16, 32, 64, 128, and 256 and the *anchor aspect ratios* were 0.2, 0.3, 0.5, 0.7, and 1. These values were chosen to favor the segmentation of unpruned shoots, which generally have a narrow and elongated shape. We performed a set of training experiments to empirically find the best learning rate for this model and found a value of 0.0001.

Table 3. Data augmentation operations used to train Mask R-CNN models.

Augmentation Operation	Values	Description
Random Horizontal Flip	0.5	Flip the image horizontally with the given probability
Random Brightness	(0.8, 1.8)	Changes the image's brightness
Random Contrast	(0.6, 1.3)	Changes the image's contrast
Random Saturation	(0.8, 1.4)	Changes the image's saturation levels
Random Rotation	[−5, 5]	Rotates the image randomly in the interval
Random Crop	(0.8, 0.8)	Randomly crop a rectangle region out of an image

In YOLOv8, the configuration was simplified by the Ultralytics framework, which provides an automatic optimizer that not only selects the type of optimizer to use from those available, but also automatically sets the learning rate, dynamics, and batch size.

To determine the appropriate batch size, YOLOv8 evaluates GPU memory availability. The framework executes a preliminary test to identify the largest batch size that utilizes approximately 60% of the available CUDA memory (https://docs.ultralytics.com/modes/train/#train-settings (accessed on 20 May 2025)). This strategy balances memory efficiency with training throughput and reduces the risk of out-of-memory errors.

Optimizer and learning rate selection are handled adaptively. In the initial training phase (first 10,000 iterations), YOLOv8 defaults to the AdamW optimizer, due to its ability to achieve faster convergence at early stages [25]. After surpassing the 10,000-iteration threshold, the optimizer automatically switches to Stochastic Gradient Descent (SGD), which generally exhibits better generalization performance in deep learning tasks.

In our specific training scenario, the dataset comprises 1288 images, partitioned into 10 folds for cross-validation (k = 10). With a batch size of 5, each fold processes approximately 1159 images, yielding around 232 iterations per epoch. Over the full training cycle, the total number of iterations exceeds 10,000, thus activating the optimizer transition mechanism. As a result, training begins with AdamW, using a learning rate of 0.001429 and momentum of 0.9, and subsequently, transitions to SGD, by YOLOv8's internal optimization logic (https://docs.ultralytics.com/reference/engine/trainer/#ultralytics.engine.trainer.BaseTrainer.build_optimizer (accessed on 20 May 2025).

Again, transfer learning was applied from the image segmentation model provided by Ultralytics pretrained on the Microsoft Common Objects in Context (COCO) dataset. Only the final layers of the network are trained, freezing the first 22 layers and training the head only. In this second group of experiments, data augmentation was also used with the operations listed in Table 4.

Table 4. Data augmentation operations used to train YOLO models.

Augmentation Operation	Values	Description
Horizontal Flip	0.5	Flip the image horizontally with the given probability
Hue	0.015	Randomly adjust the hue gain with the this maximum
Saturation	0.7	Randomly adjust the saturation gain with this maximum
Value	0.4	Randomly adjust the value gain with this maximum
Scale	0.5	Scaling factor range from 0.5 to 1.5
Translate	0.1	Maximum translation as a fraction of the image size
Erasing	0.4	Probability of randomly erasing a portion of the image

Among the different versions of YOLO, we chose version 8 in the sizes nano (N), small (S), medium (M), large (L), and extra large (X). Table 5 reflects the number of trainable parameters of each of the models used, according to the specifications of their developers.

Table 5. Parameters of the neural models used in this work (in millions of parameters).

Model	Mask R-CNN	YOLO-N	YOLO-S	YOLO-M	YOLO-L	YOLO-X
Parameters	44	3.4	11.8	27.3	46.0	71.8
GFLOPs	289	12.6	42.6	110.2	220.5	334.1
Default Input Size	640–800	640	640	640	640	640
Employed Input Size	1024	1024	1024	1024	1024	1024

For both models, we used fixed size images of 1024 pixels width. Both models were trained on 1000 epochs using *early stopping* as the regularization technique with a patience of 100 epochs.

To guarantee the statistical reliability of the results, a 10-fold cross-validation scheme was employed. The 10-fold results are illustrated using box plots, which facilitate the visual evaluation of the segmentation performance for each experiment.

All training experiments were run on a dedicated machine with Ubuntu 20.04 operating system, 2 Intel Xeon Silver 4310 CPU, 512 GB RAM (Intel, Santa Clara, CA, USA), and four Nvidia A100 PCIe 40 GB (Nvidia, Santa Clara, CA, USA).

3.3. Metrics

COCOEvaluator [26] was employed to compute segmentation metrics. We considered Precision (P), Recall (R), mean Average Precision (mAP), and Average Precision (AP) by category to measure the system performance.

Precision, as illustrated by Equation (1), is the fraction of relevant objects among the ones that were predicted. Recall, as given by Equation (2), is the fraction of relevant objects that were predicted. Precision is crucial when the cost of false positives is high and Recall is vital when the cost of false negatives is high.

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{1}$$

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2}$$

Finally, the area under the precision–recall curve (AP) can be used as a single metric to summarize the performance of the object segmentation model. The rate of TP, FP, and FN, and hence, the values of P, R, and AP, depend on the Intersection over the Union (IoU) which establishes when a detected object overlaps sufficiently with a real object. Thus, the AP50 measures the AP of the model by setting the IoU threshold at 0.5. In a multi-class object detection or segmentation task, the mAP is used, where individual AP is averaged over all classes.

The detection of the initial part of the shoots will be used to calculate the cut-off points at a later stage. These cut-off points are valid as long as they are at any position between the second and third buds. Hence, we use an IoU value of 0.5 which provides a good tradeoff between localization accuracy and robustness to small variations [27]. This ensures reliable detection of shoot segments for automated pruning.

4. Experiments and Results

This section presents the results of Mask R-CNN and YOLOv8 experiments, employing the N, S, M, L, and X versions of YOLOv8. The experiments analyze the performance of both frameworks in terms of segmentation efficiency and inference times on a Jetson AGX Orin GPU.

4.1. Segmentation

To compare segmentation performance, we used the mAP50 and AP50 per category metrics. The values of the medians of both parameters, resulting from the 10-fold cross-validation process, are shown in Table 6. To provide a clearer view of the cross-validation process, Figure 4a shows the cross-validation AP50 values for each network model, where each boxplot corresponds to the mAP50 or AP50 values obtained after the 10 runs in the K-fold cross-validation scheme.

To assess whether there exist statistically significant differences among the models performance, and considering that we have more than two groups of independent data and small sample sizes (equal to 10), we conducted a non-parametric Kruskal–Wallis test. The results yielded a p-value < 0.001, allowing us to conclude that significant differences exist between at least two groups. Subsequently, we performed pairwise comparisons using the Dwass-Steel-Critchlow-Fligner test [28] to identify which specific differences in model performance were statistically significant. The obtained p-values indicate that the

performance of the Mask R-CNN model differs significantly from all the YOLO variants, both in terms of mAP50 and AP50 across all classes. Regarding comparisons among the YOLO models, we found significant differences between YOLO-N and the other YOLO models (aside from YOLO-S), but only for the mAP50 and AP50 of the *pruned shoot* class.

Therefore, we can conclude that Mask R-CNN has a lower segmentation performance than all YOLO8 variants when considering all classes. Specifically, Mask R-CNN has an mAP50 of 0.521, while the median value for all versions of YOLOv8 is greater than 0.87. Finally, although version M shows the best global performance with an mAP50 value of 0.897, the difference from the performance of version S is not significant, so we propose to use version S as it is simpler and its inference time is the lowest.

Table 6. mAP50 and AP50 values for the studied models and classes (ten-fold cross-validation median).

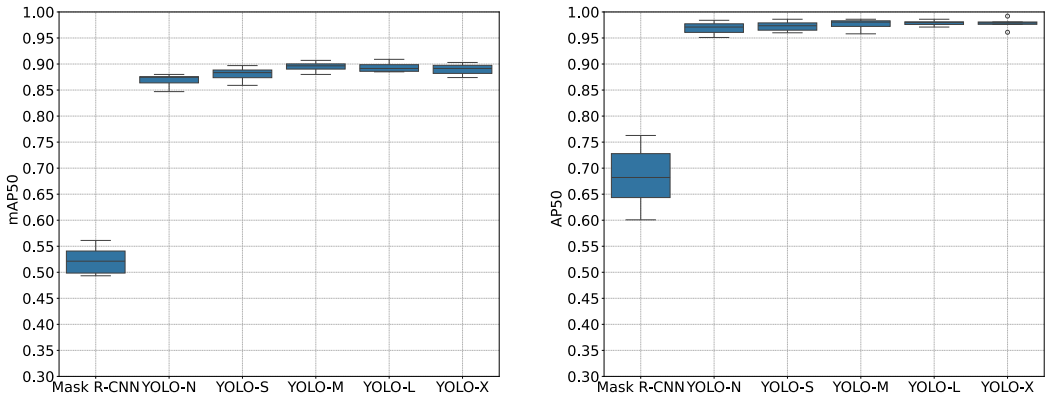
Class	Metric	Mask R-CNN	YOLO-N	YOLO-S	YOLO-M	YOLO-L	YOLO-X
All	mAP50	0.521	0.875	0.883	0.897	0.891	0.891
Trunk	AP50	0.682	0.971	0.974	0.980	0.980	0.979
Pruned shoot	AP50	0.520	0.927	0.935	0.946	0.944	0.947
Shoot	AP50	0.359	0.724	0.748	0.772	0.750	0.743

The use of a global mAP as the only metric for our study can hide segmentation errors in the most important class for the pruning task (*shoot*). The *trunk class* is only used in the post-processing phase to distinguish the lower and upper cutting points of the vine shoots, so errors in its correct segmentation are not critical. The *pruned shoot class* has been added to allow the use of the robotic pruner in vineyards where partial pruning is performed. Both an FP and FN in this class will have no effect on the management of the cutting tool. Hence, the most important errors are those that occur on the *shoot class.* An FP in this class would take the cutting tool to an area where there is nothing to cut and an FN would leave the shoot not pruned.

Therefore, it is necessary to study the AP50 values per category for each network model. Figure 4b shows AP50 behavior for the trunk class, and again, Mask R-CNN performs significantly worse than all YOLOv8 variants. In fact, the performance improvement achieved by YOLOv8 over Mask R-CNN is greater than 0.29 when comparing its median values. As there are statistically non-significant differences in the performance of the variants of YOLOv8 models in this class, we considered variant S, which achieves an AP50 of 0.974. The same considerations can also be extended to the AP50 for the *pruned shoot class* depicted in Figure 4c.

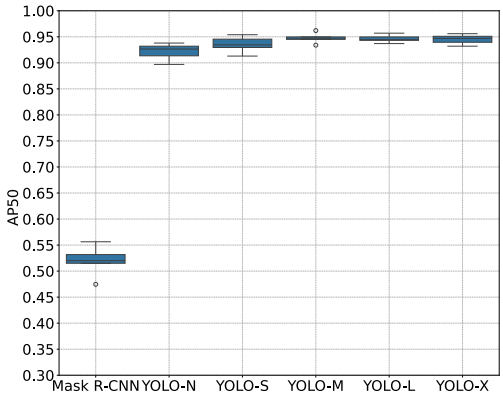
As the *shoot class* is the most important for the pruning task, a more comprehensive analysis should be conducted on this class in order to determine definitively which model is suitable for a production environment. Figure 4d shows a lower performance of all network models for this class, and Mask R-CNN also ranks last in performance with a median AP50 value of 0.359. Among the versions of YOLOv8, the differences in performance are very small (with a median AP50 value around 0.71–0.76) and are not statistically significant, so the use of the larger versions (L, M, or X), more complex and slow, is not justified. Again, YOLOv8-S seems to be the best option. In some contexts, where the training dataset is not large enough or even with correlation between the data, a smaller model such as S might offer better robustness to overfitting than larger models such as X or L, providing a better generalization ability in the data [29].

Analyzing the possible causes that lead to a lower performance of our system for the *shoot* class, compared to the other two classes, we found that the matching of the proposals to the ground-truth marks sometimes does not meet the established IoU. In our opinion, this is the most complex class, because we want the system to model a part of the object instead of the whole object, as is usual in segmentation.

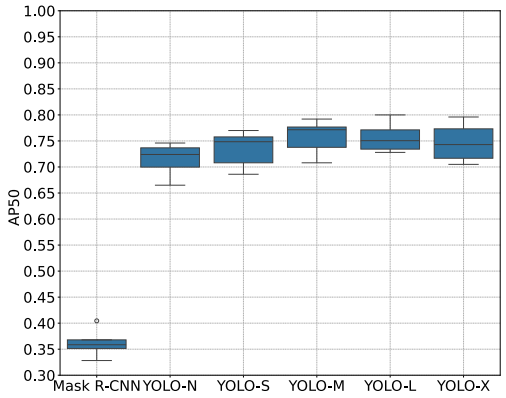


(**a**) All classes

(**b**) Trunk class



(**c**) Pruned shoot class



(**d**) Shoot class

Figure 4. mAP50 and AP50 values for the studied algorithms. (**a**) The results of the cross-validation procedure for all classes and (**b**–**d**) the AP50 results disaggregated by class.

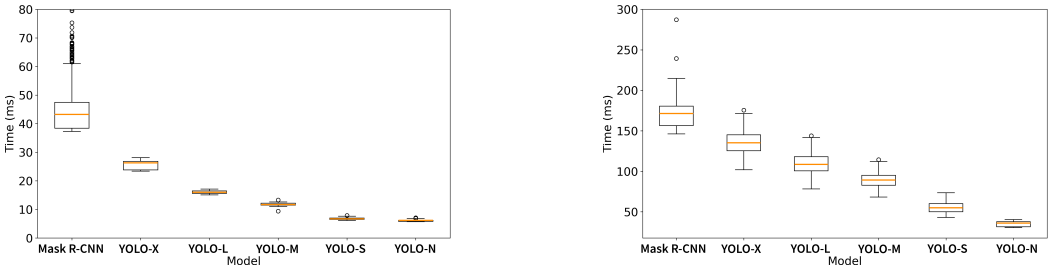
4.2. Inference Time

In the production environment, an intensive energy consumption increases the battery requirements of the robot. Therefore, it may be interesting to offload the inference process to near base station with stable power supply. The data link energy consumption will be significantly lower than on-board image processing. For this reason, two different inference tests were conducted: on the training server and on a Jetson AGX Orin.

Table 7 shows the average inference times of the studied models running on the two machines. The first test was performed on the same machine used in the training phase and Figure 5a shows the distribution of this parameter using a boxplot. On the one hand, we can observe that Mask R-CNN performs significantly worse than all YOLOv8 versions. On the other hand, YOLOv8 models have inference times ranging from 23.3 ms to 6.2 ms, which allows theoretic processing speeds from 38 to 161 frames per second (FPS) and significantly exceeds the minimum standards required for this application.

Table 7. Average inference times for the studied network models during the cross-validation process (milliseconds).

Model	GPU	Jetson AGX Orin
Mask R-CNN	43.3	171.3
YOLOv8-X	26.3	135.3
YOLOv8-L	16.0	108.6
YOLOv8-M	11.7	89.2
YOLOv8-S	6.7	55.0
YOLOv8-N	6.2	36.0

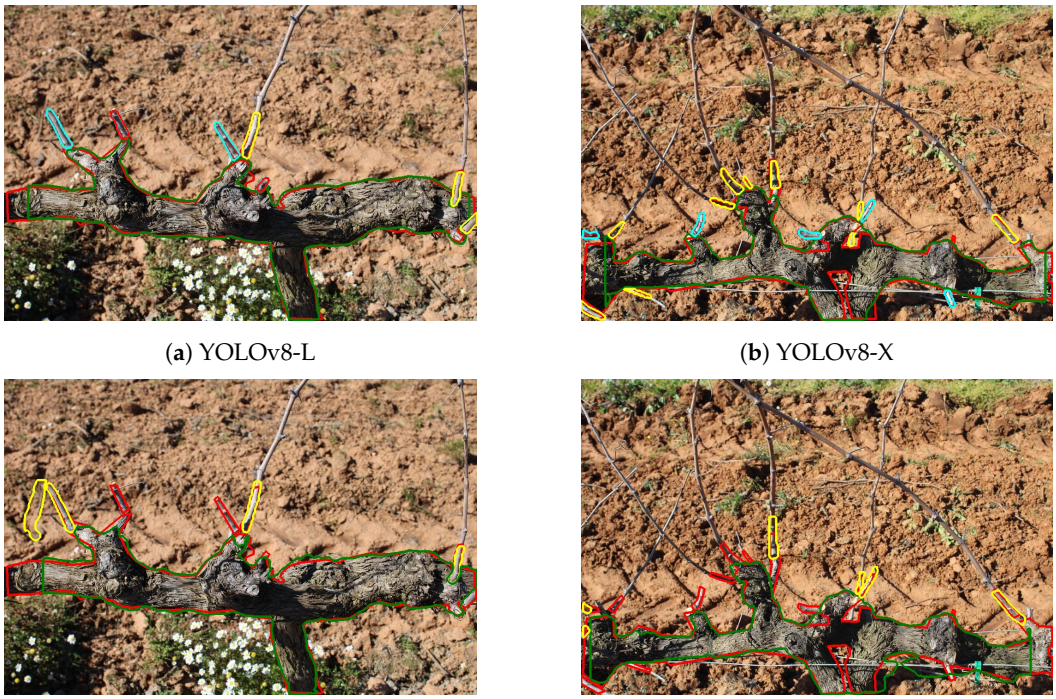


(**a**) Nvidia A100 GPU (**b**) Jetson AGX Orin

Figure 5. Inference time boxplots running on an Nvidia A100 GPU and Jetson AGX Orin.

Considering the integration and compactness objectives of the work, an integrated computing board was initially considered the optimal option for its development. For this scenario, Figure 5b shows the inference times distribution required by the network models under consideration running on a Jetson AGX Orin. First, we observe an increase in general inference times compared to those obtained on the training computer. Second, we see that the superiority of all YOLOv8 models over Mask R-CNN is repeated. Furthermore, finally, the inference time required for the YOLOv8-S network allows us to achieve a processing speed about 18 FPS.

In Figure 6, various YOLOv8 inference examples are presented, where ground truth marks are shown in red to facilitate comparison. Figure 6a shows the inference result with YOLOv8-L for the labeled image previously shown in Figure 2a. We can see that all the objects of the classes trunk and shoot are correctly detected but two FNs on the class pruned shoot are obtained. In spite of these FNs, all the objects of interest for our final purpose are correctly detected, and hence, the robotic arm would be guided to prune all the desired shoots in the image. Similarly, Figure 6b shows the results for a complex example (Figure 2b) correctly handled by one of the ten networks trained for YOLOv8-X, on an image not included in its training set. Finally, Figure 6c,d show the inference over YOLOv8-S and YOLOv8-N. Additional examples of how our system works can be found in Appendix A.



(**a**) YOLOv8-L (**b**) YOLOv8-X

(**c**) YOLOv8-S (**d**) YOLOv8-M

Figure 6. Inference examples obtained with various versions of YOLO. Ground truth marks are shown in red, inferred shoots are shown in yellow, pruned shoots are shown in blue, and inferred trunk are shown in green.

5. Conclusions

In this paper, we approached the automatic pruning of vineyards problem, using deep learning and computer vision algorithms, specifically Mask R-CNN and YOLOv8.

The results of the experiments demonstrate that YOLOv8 outperforms Mask R-CNN in the two test scenarios analyzed, both in inference time and segmentation success rate. Analyzing the results of the different YOLOv8 sizes, we can affirm that although version M shows the best global performance with an mAP50 value of 0.897, the difference from the performance of version S is not statistically significant, so we propose to use version S, as it is simpler and its inference time is the lowest.

The inference speed achieved on the Jetson AGX Orin leads us to believe that embedded deployment is feasible. In the following part of the project, we intend to drive the robot in front of the vine using RTK technology. Therefore, the inference time only affects the positioning of the cutting arm. Since the repositioning of the arm exceed 55 ms, we believe that the speed achieved is sufficient for the task.

As mentioned in Section 1.2, the purpose of several referenced papers [8,9,13] is the detection of full buds, and not to the estimation of the cutoff point. In addition, some of these works are based on other pruning techniques, such as Guyot. Hence, a direct comparison of our results with those achieved in other state-of-the-art proposals is somewhat complex.

The most important difference to highlight between our work and those analyzed in Section 1.2 lies on the attempt to directly estimate the safe pruning area in the shoot. By eliminating further post-processing, we believe that it simplifies the structure of a future robot and can increase the robustness of the system.

In conclusion, the results obtained represent a valuable advance over previous studies and open the door to future work:

- Establish a figure of merit to measure the accuracy in the cut-point identification process, test this functionality, and implement it;
- Estimate, according to the pruning density desired by the user, the shoots to be cut in the lower cutting point and those to be cut in the upper one;
- Expand the set of prototypes to ensure a correct response in any planting condition.

Author Contributions: Conceptualization, M.M.M.; methodology and software, E.P. and E.A.; validation, E.P. and M.M.M.; formal analysis, R.G. and H.M.G.V.; investigation, E.P. and M.M.M.; resources, M.M.M.; data curation, E.P.; writing—original draft preparation and writing—review and editing and visualization, R.G. and H.M.G.V.; supervision, C.J.G.-O.; project administration, M.M.M.; funding acquisition, M.M.M. and C.J.G.-O. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Innovation through the project "SISTEMA ROBOTIZADO PARA LA PODA INTELIGENTE DE VIÑEDOS (TED2021-131242B-I00)", corresponding to the 2021 Call for Projects oriented to the Ecological and Digital Transition, and GR24086 by the Junta de Extremadura and FEDER.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The set of images and their *ground truth* used in this work have been published in the open access journal *Data in Brief* under the Repository name: *VidPrune Dataset*.

Acknowledgments: The authors would like to thank Miguel González Velasco for their advice and help with the statistical analysis.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AP	Average Precision
AP50	Average Precision for an IoU threshold of 0.5
CNN	Convolutional Neural Network
COCO	Microsoft Common Objects in Context
FN	False Negative
FP	False Positive
FPS	Frames Per Second
GPS	Global Positioning System
GPU	Graphics Processing Unit
IoU	Intersection over Union
mAP	Mean Average Precision
mAP50	Mean Average Precision for an IoU threshold of 0.5
R-CNN	Region-based Convolutional Neural Network
RTK	Real-Time Kinematic
TP	True Positive
TN	True Negative
VIA	VGG Image Annotator
YOLO	You Only Look Once

Appendix A. Outputs of the Inference Process

In the following figures inferred shoots are shown in yellow, pruned shoots are shown in blue and inferred trunks are shown in green.




References

1. Jones, G.V.; Webb, L.B. Climate Change, Viticulture, and Wine: Challenges and Opportunities. *J. Wine Res.* **2010**, *21*, 103–106. [CrossRef]
2. Ryan, M. Labour and skills shortages in the agro-food sector. In *OECD Food, Agriculture and Fisheries Papers*; OECD Publishing: Paris, France, 2023. [CrossRef]
3. OECD. *Agricultural Policy Monitoring and Evaluation 2024*; OECD Publishing: Paris, France, 2024; p. 660. [CrossRef]
4. Ruzzante, S.; Labarta, R.; Bilton, A. Adoption of agricultural technology in the developing world: A meta-analysis of the empirical literature. *World Dev.* **2021**, *146*, 105599. [CrossRef]
5. Bruez, E.; Cholet, C.; Giudici, M.; Simonit, M.; Martignon, T.; Boisseau, M.; Weingartner, S.; Poitou, X.; Rey, P.; Geny-Denis, L. Pruning Quality Effects on Desiccation Cone Installation and Wood Necrotization in Three Grapevine Cultivars in France. *Horticulturae* **2022**, *8*, 681. [CrossRef]
6. Bogue, R. Robots addressing agricultural labour shortages and environmental issues. *Ind. Robot* **2024**, *51*, 1–6. [CrossRef]
7. Sze, S.; Kunze, L. Real-time 3D semantic occupancy prediction for autonomous vehicles using memory-efficient sparse convolution. In Proceedings of the 2024 IEEE Intelligent Vehicles Symposium (IV), Jeju Island, Republic of Korea, 2–5 June 2024; pp. 1286–1293. [CrossRef]
8. Botterill, T.; Paulin, S.; Green, R.; Williams, S.; Lin, J.; Saxton, V.; Mills, S.; Chen, X.; Corbett-Davies, S. A Robot System for Pruning Grape Vines. *J. Field Robot.* **2017**, *34*, 1100–1122. [CrossRef]
9. Gentilhomme, T.; Villamizar, M.; Corre, J.; Odobez, J.M. Towards smart pruning: ViNet, a deep-learning approach for grapevine structure estimation. *Comput. Electron. Agric.* **2023**, *207*, 107736. [CrossRef]

10. Lanza, B.; Nuzzi, C.; Botturi, D.; Pasinetti, S. First Step Towards Embedded Vision System for Pruning Wood Estimation. In Proceedings of the 2023 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor), Pisa, Ital, 6–8 November 2023; pp. 757–762. [CrossRef]
11. Villegas Marset, W.; Pérez, D.S.; Díaz, C.A.; Bromberg, F. Towards practical 2D grapevine bud detection with fully convolutional networks. *Comput. Electron. Agric.* **2021**, *182*, 105947. [CrossRef]
12. Chen, Z.; Wang, Y.; Tong, S.; Chen, C.; Kang, F. Grapevine Branch Recognition and Pruning Point Localization Technology Based on Image Processing. *Appl. Sci.* **2024**, *14*, 3327. [CrossRef]
13. Oliveira, F.; da Silva, D.Q.; Filipe, V.; Pinho, T.M.; Cunha, M.; Cunha, J.B.; dos Santos, F.N. Enhancing Grapevine Node Detection to Support Pruning Automation: Leveraging State-of-the-Art YOLO Detection Models for 2D Image Analysis. *Sensors* **2024**, *24*, 6774. [CrossRef] [PubMed]
14. Fernandes, M.; Scaldaferri, A.; Fiameni, G.; Teng, T.; Gatti, M.; Poni, S.; Semini, C.; Caldwell, D.G.; Chen, F. Grapevine Winter Pruning Automation: On Potential Pruning Points Detection through 2D Plant Modeling using Grapevine Segmentation. In Proceedings of the 2021 IEEE 11th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), Jiaxing, China, 27–31 July 2021; pp. 13–18. [CrossRef]
15. Pacioni, E.; Abengozar-García, E.; Macías Macías, M.; García Orellana, C.J.; Gallardo Caballero, R.; García Manso, A. Mask R-CNN aplicado a la poda de viñedos. In Proceedings of the XX Conferencia de la Asociación Española para la Inteligencia Artificial, A Coruña, Spain, 19–21 June 2024.
16. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988. [CrossRef]
17. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
18. Wu, Y.; Kirillov, A.; Massa, F.; Lo, W.Y.; Girshick, R. Detectron2. 2019. Available online: https://github.com/facebookresearch/detectron2 (accessed on 25 November 2024).
19. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788. [CrossRef]
20. Varghese, R.; Sambath, M. YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness. In Proceedings of the 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS), Chennai, India, 18–19 April 2024; pp. 1–6. [CrossRef]
21. Pacioni, E.; Abengózar, E.; Macías Macías, M.; García Orellana, C.J.; González Velasco, H.M.; García Manso, A. Vineyard dataset for automatic pruning based on main parts localization. *Data Brief* **2025**, *59*, 111335. [CrossRef] [PubMed]
22. Dutta, A.; Zisserman, A. The VIA Annotation Software for Images, Audio and Video. In Proceedings of the 27th ACM International Conference on Multimedia, Nice, France, 21–25 October 2019; Association for Computing Machinery: New York, NY, USA, 2019. [CrossRef]
23. Dablain, D.A.; Bellinger, C.; Krawczyk, B.; Chawla, N.V. Efficient Augmentation for Imbalanced Deep Learning. In Proceedings of the 2023 IEEE 39th International Conference on Data Engineering (ICDE), Anaheim, CA, USA, 3–7 April 2023; pp. 1433–1446. [CrossRef]
24. Crasto, N. Class Imbalance in Object Detection: An Experimental Diagnosis and Study of Mitigation Strategies. *arXiv* **2024**, arXiv:2403.07113.
25. Keskar, N.S.; Socher, R. Improving Generalization Performance by Switching from Adam to SGD. *arXiv* **2017**, arXiv:1712.07628.
26. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In *Proceedings of the Computer Vision–ECCV 2014*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer: Cham, Switzerland, 2014; pp. 740–755.
27. Everingham, M.; Eslami, S.M.A.; Van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes Challenge: A Retrospective. *Int. J. Comput. Vis.* **2015**, *111*, 98–136. [CrossRef]
28. Douglas, C.E.; Michael, F.A. On distribution-free multiple comparisons in the one-way analysis of variance. *Commun. Stat.-Theory Methods* **1991**, *20*, 127–139. [CrossRef]
29. Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; Vinyals, O. Understanding deep learning requires rethinking generalization. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.