Real-Time Leaves Segmentation in RGB Images with Deep Learning in a Single-Board Computer

Clodoaldo de Souza Faria Júnior[1,2], Milton Hirokazu Shimabukuro[1], Antonio Maria Garcia Tommaselli[1], Marcos Ricardo Omena de Albuquerque Maximo[3], Letícia Rosim Porto[1], Nilton Nobuhiro Imai[1],

[1]São Paulo State University (UNESP) at Presidente Prudente, São Paulo, Brazil – (clodoaldo.souza, milton.h.shimabukuro, a.tommaselli, leticia.porto, nilton.imai)@unesp.br
[2]Inspectral, Presidente Prudente, São Paulo 19060-900, Brazil – (clodoaldo.souza)@inspectral.com.br
[3]Aeronautics Technological Institute, São José dos Campos, São Paulo 12228-900, Brazil – (mmaximo)@ita.br

Keywords: Segment Anything Model, Leaf Segmentation, Deep Learning, Jetson Nano, Real-Time Application.

Abstract

This work proposed and evaluated methods for real-time leaf segmentation using a single-board computer. The main aim was to explore the state-of-the-art techniques based on the YOLO algorithm for real-time operation. For this purpose, the available variants of YOLOv8 and YOLOv9 were evaluated, and a semi-automatic labelling method based on the Segment Anything Model (SAM) algorithm was used. Given the need to delimit the leaf contour for labelling, it was possible to create a larger and more accurate dataset compared to the purely manual procedure. In addition, the cost-benefit of the applied algorithms and methods were assessed, considering the computational demand required, as well as the accuracy, recall, and precision delivered by these techniques. In this study, both quantitative analysis of the trained architectures' metrics and qualitative examination through direct observation of images were conducted to identify crucial aspects. The experiments were conducted with a post-processed dataset and the suitability for real-time applications was based on the elapsed time for segmentation. We concluded that the YOLOv8n architecture is the best one among those tested, presenting a precision and recall of 0.9064 and 0.7233, respectively. This architecture represents the best cost-benefit ratio between computational cost and real-time performance, being able to perform segmentation in 310 ms with the NVIDIA Jetson Nano board. Furthermore, when computational cost is not a problem or even when segmentation time can be higher, the YOLO8m network may be recommended when the recall metric is more important than precision. This network presented a precision and recall of 0.8556 and 0.7726, respectively, and presented a better performance in segmenting leaves located in more complex parts of the image and with a higher recall.

1. Introduction

Promoting sustainable agriculture, optimizing crop yield and plant phenotyping are crucial needs. Characteristics such as colour, shape, plant height, leaf area index, and growth rate are essential data for the phenotypic assessment. In this regard, the automated extraction of leaves from plant images can significantly boost phenotypic analysis with the advantage that it is a non-destructive technique (Li et al., 2014). With this approach, it becomes feasible to monitor the growth cycle of crops, identify plant health problems, and optimize agricultural practices (Ghazal et al., 2019). A particular problem is the automatic leaf segmentation in images of plants with complex backgrounds.

The advances in image processing technology and deep learning algorithms have enabled several researches to identify and segment multiple leaves in a single image. Aich and Stavness (2017) employed a deep-learning architecture to count leaves in plant images. Kuznichov et al. (2019) enhanced leaf segmentation accuracy using data augmentation methods. In a comparative study, presented by Scharr et al. (2016), four leaf segmentation methods for digital plant images using deep learning were evaluated, resulting in an average accuracy exceeding 90%. However, researchers noted that a complex background could affect this accuracy. To cope with these challenges, Yang et al. (2020) proposed leaf image segmentation and classification with complex backgrounds using Mask R-CNN (Mask Region Convolutional Neural Network) (He et al., 2017) to recognize and extract object regions from the background at the pixel level. According to the authors, this method is suitable for leaf segmentation, having fewer parameters and lower depth architecture.

However, in real-world applications such as agricultural fields, small and low-latency models are often required, explicitly tailored for devices with limited memory and computational power while maintaining comparable or better accuracy. In this context, Cao et al. (2023) proposed the use of Mask R-CNN for the detection and segmentation of strawberries in an orchard using images. This model combines features from different scales and was implemented on a Jetson Nano board, achieving 19 frames per second (FPS) and a mean average precision (mAP) of 79.7%. Liu et al. (2023) proposed the use of a model called NanoSegmenter, based on the Transformer structure, for segmentation and disease detection in a tomato plant. This model can run on Jetson Nano, achieving an inference speed of 37 FPS with a precision of 98%. Additionally, Ji et al. (2022) proposed a detection and localization method for a harvesting robot based on Shufflenetv2-YOLOX, built upon YOLOX-Tiny in conjunction with Shufflenetv2. This method achieved an average precision of 96.76% and a detection speed of 65 FPS. These models can be efficiently deployed in the field using single-board computing (SBC) platforms like Jetson Nano, ensuring practical applicability in real-world agricultural environments as mentioned in Assunção et al. (2022).

Hence, it is apparent that current studies in this field tend to favour simpler architectures to attain optimal performance in frames per second (FPS), often relying on outdated and streamlined neural network architectures. Given this perspective, the contribution of this work involves using deep learning networks for real-time implementation and assessing their viability for deployment on the Jetson Nano board. This method will be utilized to perform leaf instance segmentation in terrestrial images captured under ambient lighting conditions, thereby addressing variations induced by natural illumination and object occlusions commonly faced in natural scenes. These

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-3-2024-139-2024

scenes often involve partial occlusions, shadows, and notable differences in illumination levels.

The remainder of the paper is organized as follows: in Section 2, the theoretical foundations regarding the artificial intelligence methods that were investigated are presented, as well as the single board computer (SBC) that was used for the application of the trained methods. In Section 3, the materials and methods used in the present study are presented, including the libraries, programming language, and configurations employed to achieve the presented results. In Sections 4 and 5, the results and conclusions are presented, respectively.

2. Background

2.1 YOLOv8

YOLO (You Only Look Once) was originally an object detection method designed to meet various demands, aiming to locate and label objects directly in a single pass through the network, enabling satisfactory performance in object detection in general images (Adibhatla et al., 2020). However, over the years variations and adaptations have emerged to use YOLO in other tasks than detection, such as segmentation. Therefore, a series of YOLO network versions have been widely used in the industry, including the latest variants YOLOv8 (Jocher et al., 2023) and YOLOv9 (Wang et al., 2024). YOLOv8 was released in January 2023 by Jocher et al. (2023).The version v8 has five different scales, i.e., network depth and number of neurons and parameters: YOLOv8n (nano), v8s (small), v8m (medium), v8l (large), and v8x (extra-large). YOLOv8 supports various computer vision tasks such as object detection, segmentation, pose estimation, tracking, and classification. Despite the different names and applications, obtaining any of these outputs requires passing through the network only once, making it a single-step process.

Figure 1 illustrates the detailed architecture of YOLOv8, which employs a backbone similar to YOLOv5, with some changes in the CSPLayer, now called the C2f module. This module combines high-level features with contextual information to enhance detection accuracy. Additionally, YOLOv8 utilizes an anchor-free model with a decoupled head to process detection, classification, and regression tasks. This design allows each branch to focus on its respective task, contributing to improving the overall model accuracy (Jocher et al., 2023).

In the output layer of YOLOv8 (Head), the sigmoid function is adopted as the activation function for the score, which represents the probability that the bounding box contains a given object. Additionally, the softmax function is applied to calculate class probabilities, indicating the likelihood of an object belonging to a certain class. Finally, the Dynamic Focal Loss (DFL) and Completed Intersection over Union (CIoU) loss functions are used for bounding box loss, while binary cross-entropy is employed for classification or segmentation loss, being that the cross-entropy calc will change depending on the network application. These losses are chosen to enhance object detection performance, especially in cases involving smaller objects (Terven et al., 2023).

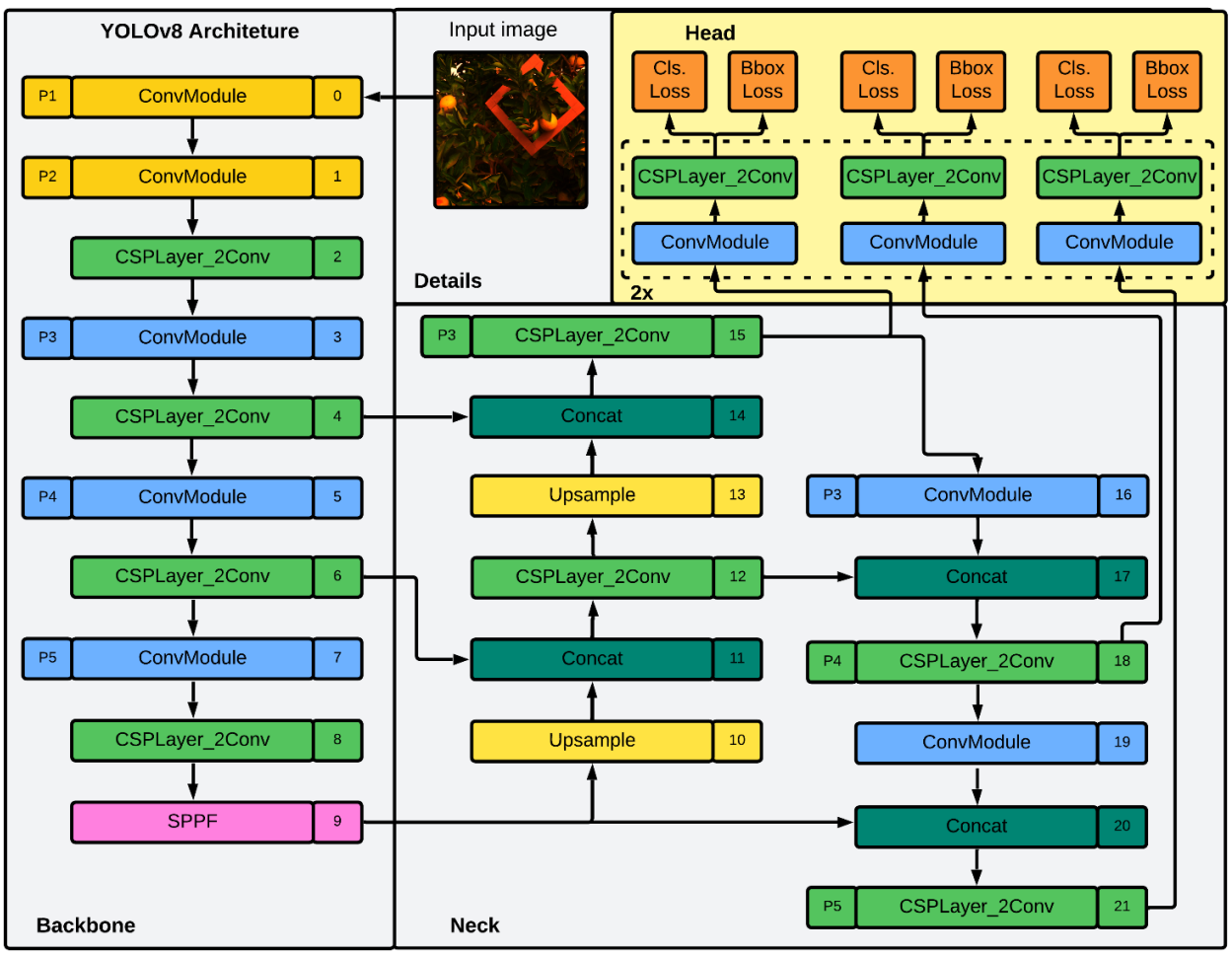

Figure 1. Architecture of YOLOv8. Source: Adapted from Jocher et al., (2022).

2.2 YOLOv9

On the other hand, YOLOv9 incorporates the concept of Programmable Gradient Information (PGI), aimed at addressing the challenge of data loss in deep neural networks. In traditional architectures, as information passes through various layers, there are some losses, resulting in less efficient learning and model performance degradation. PGI allows for more precise control over gradients during the training process, ensuring that critical information is preserved and used more effectively. This results in improved learning outcomes and model accuracy (Wang et al., 2024).

Another important concept is the use of Generalized Efficient Layer Aggregation Network (GELAN), which enhances model performance and efficiency by optimizing how different layers of the network aggregate and process information. The primary goal of GELAN is to maximize parameter utilization, ensuring that the model achieves higher accuracy without a proportional increase in computational resources or model size. This enables handling object detection tasks with greater precision and efficiency (Wang et al., 2024).

As for the available versions of YOLOv9, there are also five scales, namely: YOLOv9-n (nano), YOLOv9-s (small), YOLOv9-m (medium), YOLOv9-c (compact), and YOLOv9-e (extended), each varying the number of parameters (complexity of the model) and consequently the performance. These models cater to diverse requirements, ranging from lightweight applications to more extensive and high-performance applications. Figure 2 provides a visualization of the comparative performance between YOLOv8, YOLOv9, and the latest models released for the MS COCO dataset. The performance of the models is presented using the Average Precision (AP) metric, i.e., the fraction of correct predictions (true positives) among all predictions made by the model for MS COCO dataset. Furthermore, these values are presented using a standard video card, thus allowing a comparison of the performance of each architecture.

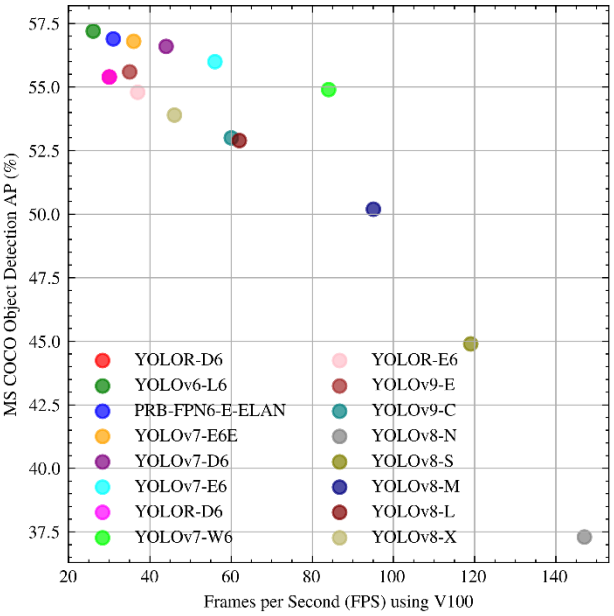

Figure 2. Performance of the latest models released for object detection running on an NVIDIA V100 graphics card, including YOLOv8 and YOLOv9.

2.3 Single Board Computers

There are numerous embedded devices that can be used in various applications, depending on the function they perform, their complexity, and supported technology, among other factors. Due to the rapid evolution of technology in recent decades, it has been possible to integrate most of the functional elements of an electronic system into a single chip. Thus, Single Board Computers (SBCs) emerge as complete computers built on a single printed circuit board that contains memory, processor, input/output devices, and other components. They are based on System-on-a-Chip (SoC), which has all the integrated components (Murshed et al., 2021).

The main function of these devices is to reduce device size and costs and increase efficiency and performance. The key components included in these devices are CPU, RAM memory, input and output controllers, Graphics Processing Unit (GPU), communication controllers, and, in some cases, Tensor Processing Unit (TPU). In general, SoCs do not have an operating system embedded in the chip itself; instead, the system that runs at initialization on the SoC’s CPU loads the operating system to the memory when the device is powered on or rebooted (Garcia-Perez et al., 2023).

3. Material and Methods

3.1 NVIDIA Jetson Nano

The NVIDIA Jetson Nano, a Single-Board Computer, was chosen for the current application due to its ability to provide significant computational power in a compact and energy-efficient form factor. The decision was strongly influenced by the presence of the NVIDIA Maxwell GPU, which enables efficient execution of deep learning models such as YOLO, which are used for real-time object segmentation. This hardware integration offers a powerful and efficient solution to meet the demands of real-time artificial intelligence processing.

3.2 Dataset and Preprocessing

In this study, the main component used for image acquisition was the multispectral camera, Sony A7R Multispectral Sextuple model, equipped with a six-lens system developed by Agrowing. This system has different lenses to produce six distinct quadrants, each one with different spectral bands. Table 1 presents the specifications of the sensor used, according to (Agrowing, 2024).

Parameter	Specification
Type	Single assembly with six lenses
Field of view	Diagonal 40.8º Horizontal 25.0º Vertical 25.0º
Lens distortion	< 2%
Multispectral bands	Q1 (405, 570, 710nm) Q2 (525, 630nm) Q3 (850nm) Q4 (430, 550, 650nm) Q5 (490, 732nm) Q6 (450, 560, 685nm)
Focal length	21.8mm
Aperture	Fixed at F/6
Dimensions	64.5x35 mm

Table 1. Specifications of the Multispectral Camera, Sony A7R Multispectral Sextuple model.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-3-2024-139-2024 | © Author(s) 2024. CC BY 4.0 License.

Regarding the dataset, ground acquisition was performed manually with the Sony A7R Multispectral Sextuple camera without the use of a tripod or any type of support. The collected data is a citrus orchard located in the countryside of São Paulo state, in the municipality of Matão (Lon. -48. 3665 and Lat. -21.6034).

The image processing was carried out using the Agrowing Basic software. The individual spectral bands are extracted with this software. Originally, the image contained information regarding six lenses, and through the initial processing, it was possible to split the image into 14 distinct images corresponding to specific wavelengths.

In this study, RGB compositions were used as input for the models to be close to those contexts of real-time applications with ordinary onboard cameras. An important aspect was to adjust the contrast of the images to improve visualization and understanding of the image contents, thereby facilitating the targets’ distinction. To achieve this, a brief study of possible methodologies was conducted, and among them, histogram stretching using the 2% and 98% percentiles was chosen (Langarizadeh et al., 2011). Thus, the process was carried out for image bands corresponding to 430, 550 and 650 nm wavelengths.

To perform histogram stretching, we first identified the lowest and highest pixel values in the image, denoted as c and d, respectively. Then, each pixel P is scaled using Equation 1. P_{out} is the normalized pixel value, a and b are the two extreme gray values, for example, in an 8-bits image, these are 0 and 255 (Jensen, 2015). Additionally, image cropping was performed to enhance the visualization and avoid the detection, classification, and segmentation over the original full-size image.

$$P_{out} = \left(\frac{P - c}{d - c}\right) \times (b - a) + a \quad (1)$$

This step, when carried out in real-time on the NVIDIA Jetson Nano, will be used as a calibration step for the image acquisition parameters.

3.3 Artificial Intelligence Frameworks, Data Collection, and Conducted Experiments

The YOLO approach was adopted as the main technique. For the implementation and execution of YOLO, the Ultralytics library was used, recognized for its efficient implementations and pre-trained models, which provide accurate and reliable object segmentation results. In addition to Ultralytics library, the PyTorch (Paszke et al., 2017, 2019) and TorchVision libraries were used. PyTorch is an open-source deep learning framework that offers flexibility and ease of use in AI model research and development. TorchVision is a complementary library to PyTorch, providing common computer vision datasets and models, as well as image transformations for data preprocessing.

After the object detection and segmentation, a thorough evaluation was necessary to ensure the quality of the results. For this analysis, the Pandas (McKinney, 2010) library was employed, known for its flexibility and data manipulation capabilities. Through Pandas, we could examine and visualize the YOLO model's output data, allowing for a deeper understanding of the detection and segmentation, and the identification of code snippets for improvement. Additionally, the NumPy (Harris et al. 2020) library played a fundamental role in various steps of the process.

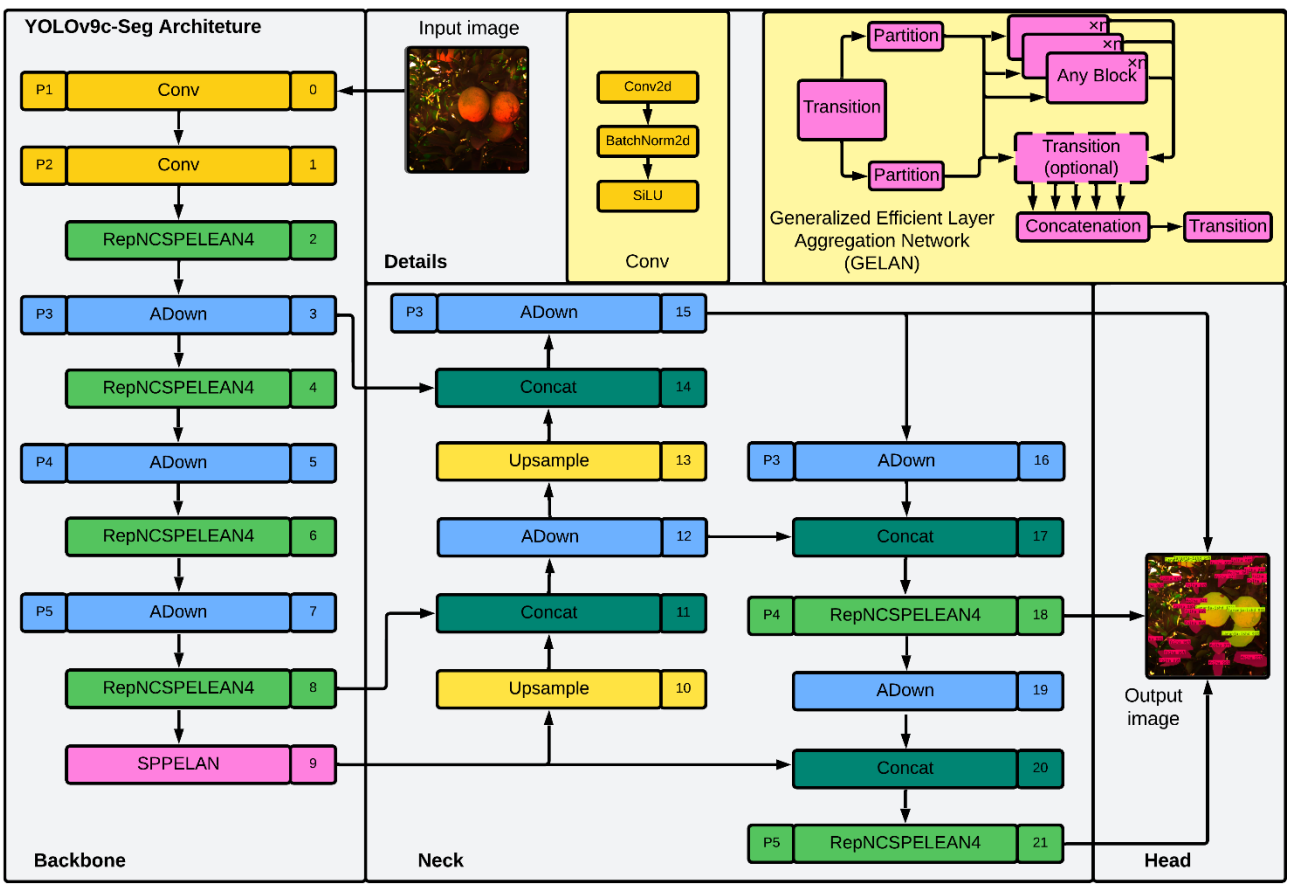

Figure 4. YOLOv9c-Seg architecture used. Source: Adapted from Wang et al., (2024)

Based on this, the dataset was created using the collected data. This step was conducted through a semi-automatic collection methodology with the assistance of the Segment Anything Model (SAM) algorithm (Kirillov et al., 2023). SAM is an image segmentation model that allows for immediate segmentation of desired features to serve as training data. Subsequently, the segments created by SAM were visually evaluated to verify if they presented satisfactory results or if they needed manual refinement. This semi-automatic labelling process provided a significant gain in time for dataset generation for training, validation, and testing. While the fully manual process is burdensome, there is still a step of manual evaluation and adjustment if necessary. Using this method, we obtained 1203 leaf segments used for training, validation, and testing, distributed in 60%, 20%, and 20%, respectively.

It is important to highlight that the SAM algorithm is used solely to construct the dataset for building the models, and it is executed only in the training environment. The training was conducted on the Google Colab platform, and only the final model was transferred to the NVIDIA Jetson Nano. Therefore, only inference is performed on the SBC. The training environment consists of an Intel Xeon Platinum, an NVIDIA T4, and 25 GB of RAM.

Although Ultralytics has already published the YOLOv9-Seg model, at the time this paper was being written, it was not possible to train it using their library. Therefore, it was necessary to manually construct the architecture for training. The architecture used to conduct the experiments is presented in Figure 4. Notice that this architecture was built based on YOLOv9c, and attempts were also made to construct the YOLOv9e variation. However, due to the lack of documentation on the latter, a variation was proposed for training execution. The proposal involves using information at the P3, P4, and P5 extraction levels, i.e., in multi-resolution, as often employed in YOLO. This information was extracted from layers 15, 18, and 21. Additionally, Figure 4 presents the generic block of GELAN, used as the basis for the proposed architecture, along with a convolutional block consisting of a set of operations. Thus, it became feasible to use the YOLOv9c architecture to perform all necessary processes, including detection, classification, and realising instance segmentation.

4. Results

Tests were carried out on each of the YOLOv8 and YOLOv9 architecture variations to select those that would be viable for real-time inference, that is, the time that each of these architectures would take to perform inference once the data are collected. Average elapsed times are presented in Table 2. It was observed that as the complexity (number of parameters) of the architecture increased the processing time also increased proportionally. In addition, despite the increased complexity of the YOLOv9-based versions, there was no significant increase in inference time, which was equivalent to that of the YOLOv8 versions.

Model	Detection	Segmentation (Detection included)	Parameters (Complexity)
YOLOv8n	220ms	310ms	3M
YOLOv8s	482ms	645ms	11M
YOLOv8m	1086ms	1385ms	27M
YOLOv8l	1773ms	2427ms	45M
YOLOv8x	2939ms	3455ms	71M
YOLOv9c	1222ms	1685ms	27M
YOLOv9e	2493ms	3131ms	59M

Table 2. Average inference times for each of the architectures running on the Jetson Nano.

Analysing the results, it is notable that any of the architectures can be used if the processing time can exceed 3455 ms. In such cases, the smallest architecture can be used, enabling execution in about 310 ms. Therefore, these architectures are suitable for real-time implementation, with the main difference between them being the inference quality and the amount of training data required. Table 3 presents the average confidence (probability) levels, mAP50, and mAP50-95, and the number of measurements (objects) achieved. The mAP50 (mean Average Precision 50% confidence) is a metric used to evaluate the accuracy of object detection models. It measures the average precision of detection across all classes when the confidence threshold is set to 50%. mAP50-95 (mean Average Precision between 50% and 95% confidence) is similar to mAP50 but considers the precision across a range of confidence thresholds, from 50% to 95%. This metric provides a broader understanding of the model's performance across different confidence levels, capturing its robustness and reliability in various scenarios. It is important to note that the images used have dimensions of 2252×2252 and are divided into several 640×640 patches to enable proper execution and feature extraction. Therefore, the construction of patches without overlapping between them and the excess part is discarded. The confidence and quantities presented in Table 3 are based on three different image examples, which are images from the test dataset Predictions are then performed on the test data and the metrics presented in the table are obtained from those predictions.

Model	mAP50	mAP50-95	Conf. (mean)	Measurements (leaves identified)
8n	0.6825	0.4106	0.7551	32
8s	0.7150	0.4443	0.7284	27
8m	0.7465	0.4589	0.6922	25
8l	0.7002	0.4084	0.7226	26
8x	0.3703	0.1612	0.8891	95
9c	0.4475	0.2127	0.2999	17
9e	0.6601	0.3919	0.6214	20

Table 3. Metrics obtained during the tests of the YOLO architectures.

The comparative analysis between the YOLOv8 and YOLOv9 architectures shows a pattern of progressive increase in segmentation time as the architecture complexity rises, as expected. For instance, the segmentation time of YOLOv8n was 220 ms, while of YOLOv8x reached 2939 ms, representing a substantial increase of approximately 1236%. However, it is worth noting that, for our dataset, this increased complexity does not necessarily translate into proportionally better performance in terms of precision metrics. For example, although YOLOv8x has the longest segmentation time, its mAP50 only reaches 37.03%, compared to YOLOv8m, which achieves a mAP50 of 74.65%, with a segmentation time of 1086 ms, representing a gain of approximately 101.62%. Regarding precision, for both mAP50 and mAP50-95, it is evident that YOLOv8m stands out, presenting the best performance with a mAP50 of 74.65% and a mAP50-95 of 45.89%. On the other hand, YOLOv8x achieved the lowest scores, with a mAP50 of 37.03% and a mAP50-95 of only 16.12%. Compared with YOLOv8n, we observed that YOLOv8m represented a gain of approximately 9.20% in mAP50 and 11.39% in mAP50-95, highlighting its superiority in

terms of object segmentation precision. Regarding the confidence of predictions, it is interesting to note that, despite the high confidence of YOLOv8x (confidence of 88.91%) its overall precision is considerably lower compared to YOLOv8m, which has a confidence of 69.22% and offers more consistent performance in terms of mAP50 and mAP50-95. The analysis suggests that YOLOv8m is the best choice in terms of object segmentation precision, as it presents a favourable balance between segmentation time and performance of precision metrics. However, if segmentation speed is the priority, YOLOv8n or YOLOv8s can be considered, although there is a precision loss.

In addition, Figure 5 shows the training and validation metrics over the epochs for each architecture. It can be observed that as the model complexity increases (as shown in Table 2), more epochs are required for training and more computational resources are used. The training step exhibits a noisy behaviour, i.e., convergence based on loss is achieved, but accuracy and recall do not fully stabilize over time. This can be justified since a small dataset was used. Even though, the results are satisfactory and can be used. The YOLOv8n (blue line) and YOLOv8s (red dashed line) networks, on the other hand, show a reasonable performance, and the YOLOv8m (green dashed line) architecture achieves the best results. This architecture presents high precision and recall values, but also the smallest possible architecture in relation to the number of parameters and layers. During training, we made an intentional choice to prioritize precision over recall. This decision was driven by the need to ensure more accurate and reliable predictions from the model. As a result, recall values tend to be lower, as expected and modelled, as it can be seen in Table 4. This emphasis on precision is particularly crucial in our current context, where false positives could have significant consequences, especially in agricultural applications where misidentification might impact critical crop management decisions.

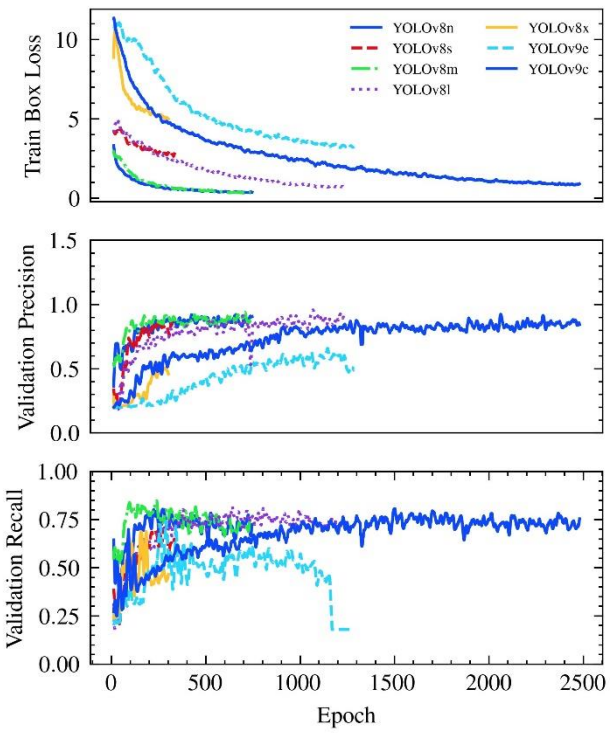

Figure 5. Precision and recall curves obtained during the validation and loss during the training of each of the models.

Model	Precision	Recall	Number of epochs
8n	0.9064	0.7233	745
8s	0.8212	0.6454	944
8m	0.8556	0.7726	738
8l	0.8842	0.7575	1235
8x	0.4596	0.4668	302
9c	0.4894	0.1800	1286
9e	0.8440	0.7545	2480

Table 4. Precision, recall, and number of epochs needed for training, obtained during the training and validation of the proposed architectures using Early Stopping.

However, if the goal is to run in the shortest possible time, the YOLOv8n architecture has approximately 0.7233 percentage points of recall but surpasses 77% in execution time, going from 1385 ms with YOLOv8m to 310 ms with YOLOv8n. It is important to note that although YOLOv8x is the most robust of all and presents the best performance when analysed on the COCO dataset (Lin et al., 2014), it requires a huge volume of data that was not available in this work. Observing the losses over the epochs, this architecture neither converged nor achieved a good result, as depicted in Figure 5. Regarding the YOLOv8s architecture, although it is intermediate between YOLOv8n and YOLOv8m, it presented a worse result for both recall and precision. Therefore, it can be concluded that increasing the architecture parameters was not beneficial in this study case. In addition, Table 4 shows the precision and recall values considered for each model using the Early Stopping method.

When analysing the YOLOv9-based models, it is observed that there is a significant increase in the number of epochs required for training. Even though the developed model presented a satisfactory result. Nevertheless, when compared to YOLOv8l, for example, there is a decrease of 4, 1, and 10 percentual points for mAP50, mAP50-95, and confidence, respectively. In addition, there is an increase of approximately 100% in training time and 29% in execution time on the NVIDIA Jetson Nano. Therefore, it is justified that for the present dataset, it is better to use the YOLOv8 architectures, since they present a shorter response time, which is suitable for real-time inference, and slightly superior performance. Finally, in Figure 6, there is an example that includes leaves that are straightforward to segment and leaves that, due to the complex scenario, are hard to segment. Figure 6 shows the bounding boxes from the detection process, the segments obtained (red masks on objects detected) for each, and the confidence scores for each of these. With these results, it is possible to verify the performance of each of the models. The best results were achieved with YOLOv9, which identified more leaves, but these results are often incorrect, as shown in Table 3, where 95 leaves were measured with a high confidence level when compared to the other models. However, it is possible that these are false negatives, based on Table 4, due to the low recall value of this model. However, for the other models, the result, despite presenting different behaviours for each one, highlights that the best architectures were YOLOv8l and YOLOv8n, as previously mentioned. With respect to YOLOv9c, it can be noted that it was able to segment some leaves, but the performance was not comparable to YOLOv9e. In addition, it is noted that the training of this model ends up having a different behaviour; after approximately 1300 epochs, the accuracy of the model tends to improve, but the recall starts to drop, reaching 0.1800, as can be seen in Table 4.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-3-2024-139-2024 | © Author(s) 2024. CC BY 4.0 License.

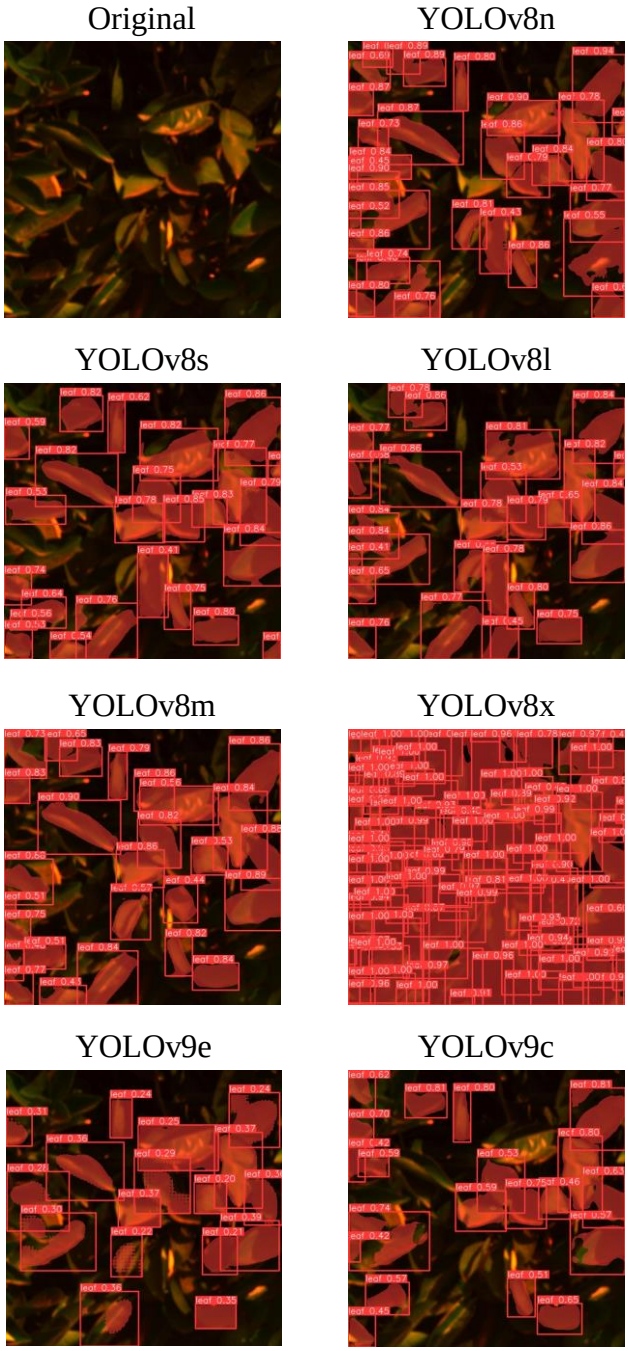

Figure 6. Bounding boxes, segments (masks) and confidence levels obtained from predicting architectures for a test image.

5. CONCLUSION

This study introduced and evaluated techniques for real-time leaf detection and instance segmentation with inferences running on a single-board computer. The primary objective was to investigate cutting-edge approaches utilizing the YOLO algorithm and its real-time capabilities. To achieve this, various YOLOv8 and YOLOv9 scaled models were examined, alongside employing a semi-automatic labelling approach based on the Segment Anything Model (SAM) technique. By focusing on delineating leaf outlines, a method was devised to generate more extensive and precise datasets compared to conventional methodologies, particularly concerning labelling. Furthermore, the study assesses the cost-effectiveness of the implemented techniques, considering their computational requirements and the confidence, recall, and precision they provide.

The models built based on the YOLO v8 and v9 architectures were experimentally assessed, and some metrics were presented and analysed. Although the inference time can reach 3455 ms, real-time applications in agriculture are viable, as confirmed in the experiments. It was also concluded that the YOLOv8 variant architectures have presented slightly superior performance when compared to the YOLOv9-based architectures, probably due to the amount of data used. It is also observed that the dataset, even having more than 1200 segments, is still considered small, compared to datasets required for training deep learning models. Therefore, for future work, it is recommended to augment the dataset.

This work proposes a large-scale and real-time data collection technique for obtaining phenotypic characteristics. With this technique, it is possible to monitor the growth cycle of crops, identify plant health problems early on, and optimize agronomic management, such as irrigation, fertilization, and pest control. In addition, it is noteworthy that this technique can contribute to the mapping of vegetation cover, biomass, productivity estimation, and weed identification. Although this work has focused on utilizing a citrus orchard, the proposed techniques for real-time leaf detection and instance segmentation are applicable to various other tree types, as they involve the extraction of leaf information. Consequently, the presented method has potential to support the estimation of the production of many other fruits, while also serving as a foundation for extending into other computer vision methodologies. It is recommended for future work to use the other bands of the Agrowing camera since this study only used the bands of the visible spectrum. It is also important to study issues such as the SBC system's onboard power supply and continuous operation time.

Acknowledgements

The authors thank The São Paulo Research Foundation (FAPESP) [Grant number: 21/06029-7], the Brazilian Federal Agency for Support and Evaluation of Graduate Education (CAPES) [Grant numbers: 8888.821780/2023-00, 88887.839524/2023-00 and 88887.817757/2023-00] and the National Council for Scientific and Technological Development (CNPq) [Grant numbers: 303670/2018-5 and 308747/2021-6].

References

Adibhatla, V.A.; Chih, H.-C.; Hsu, C.-C.; Cheng, J.; Abbod, M.F.; Shieh, J.-S. 2020. Defect detection in printed circuit boards using you-only-look-once convolutional neural networks. *Electronics* 9: 1547.

Agrowing Development Team. 2020. https://agrowing.com/. (15 Jan. 2024).

Aich, S.; Stavness, I. 2017. Leaf counting with deep convolutional and deconvolutional networks. *Proceedings of the IEEE international conference on computer vision workshops*: 2080–2089.

Assunção, E.; Gaspar, P.D.; Mesquita, R.; Simões, M.P.; Alibabaei, K.; Veiros, A.; et al. 2022. Real-time weed control application using a jetson nano edge device and a spray mechanism. *Remote Sensing* 14: 4217.

Cao, L.; Chen, Y.; Jin, Q. 2023. Lightweight Strawberry Instance Segmentation on Low-Power Devices for Picking Robots. *Electronics* 12: 3145.

Garcia-Perez, A.; Miñón, R.; Torre-Bastida, A.I.; Zulueta-Guerrero, E. 2023. Analysing Edge Computing Devices for the Deployment of Embedded AI. *Sensors* 23: 9495.

Ghazal, M.; Mahmoud, A.; Shalaby, A.; El-Baz, A. 2019. Automated framework for accurate segmentation of leaf images for plant health assessment. *Environmental monitoring and assessment* 191: 491.

Harris, C.R.; Millman, K.J.; Walt, S.J. van der; Gommers, R.; Virtanen, P.; Cournapeau, D.; et al. 2020. Array programming with NumPy. *Nature* 585: 357–362.

He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. 2017. Mask r-cnn. *Proceedings of the IEEE international conference on computer vision*: 2961–2969.

Jensen, T.J. 2015. Image Processing. In: *Budget Astrophotography*, Springer, p.57–92.

Ji, W.; Pan, Y.; Xu, B.; Wang, J. 2022. A real-time apple targets detection method for picking robot based on ShufflenetV2-YOLOX. *Agriculture* 12: 856.

Jocher, G.; Chaurasia, A.; Qiu, J. 2023. Ultralytics YOLOv8. .
Jocher, G.; Chaurasia, A.; Stoken, A.; Borovec, J.; NanoCode012; Kwon, Y.; et al. 2022. ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation.

Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; et al. 2023. Segment anything. *Proceedings of the IEEE/CVF International Conference on Computer Vision*: 4015–4026.

Kuznichov, D.; Zvirin, A.; Honen, Y.; Kimmel, R. 2019. Data augmentation for leaf segmentation and counting tasks in rosette plants. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*: 0–0.

Langarizadeh, M.; Mahmud, R.; Ramli, A.R.; Napis, S.; Beikzadeh, M.R.; Rahman, W. 2011. Improvement of digital mammogram images using histogram equalization, histogram stretching and median filter. *Journal of medical engineering & technology* 35: 103–108.

Li, L.; Zhang, Q.; Huang, D. 2014. A review of imaging techniques for plant phenotyping. *Sensors* 14: 20078–20111.

Lin, T.-Y.; Maire, M.; Belongie, S.J.; Bourdev, L.D.; Girshick, R.B.; Hays, J.; et al. 2014. Microsoft COCO: Common Objects in Context. *CoRR* abs/1405.0312.

Liu, Y.; Song, Y.; Ye, R.; Zhu, S.; Huang, Y.; Chen, T.; et al. 2023. High-Precision Tomato Disease Detection Using NanoSegmenter Based on Transformer and Lightweighting. *Plants* 12: 2559.

McKinney, W. 2010. Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*: 56–61.

Murshed, M.S.; Murphy, C.; Hou, D.; Khan, N.; Ananthanarayanan, G.; Hussain, F. 2021. Machine learning at the network edge: A survey. *ACM Computing Surveys (CSUR)* 54: 1–37.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; et al. 2017. Automatic differentiation in PyTorch. *NIPS-W*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., p.8024–8035.

Scharr, H.; Minervini, M.; French, A.P.; Klukas, C.; Kramer, D.M.; Liu, X.; et al. 2016. Leaf segmentation in plant phenotyping: a collation study. *Machine vision and applications* 27: 585–606.

Terven, J.; Córdova-Esparza, D.-M.; Romero-González, J.-A. 2023. A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction* 5: 1680–1716.

Wang, C.-Y.; Yeh, I.-H.; Liao, H.-Y.M. 2024. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. *arXiv preprint arXiv:2402.13616*.

Yang, K.; Zhong, W.; Li, F. 2020. Leaf segmentation and classification with a complicated background using deep learning. *Agronomy* 10: 1721.