agriculture

MDPI

Article

Comprehensive Analysis of Model Errors in Blueberry Detection and Maturity Classification: Identifying Limitations and Proposing Future Improvements in Agricultural Monitoring

Cristhian A. Aguilera [1,*], Carola Figueroa-Flores [2], Cristhian Aguilera [3] and Cesar Navarrete [3]

[1] Facultad de Ingeniería, Arquitectura y Diseño, Universidad San Sebastián, Lago Panguipulli 1390, Puerto Montt 5501842, Chile
[2] Departamento de Ciencias de la Computación y Tecnologías de la Información, Facultad de Ciencias Empresariales, Universidad del Bío-Bío, Chillan 3800708, Chile; cfigueroa@ubiobio.cl
[3] Departamento de Ingeniería Eléctrica y Electrónica, Facultad de Ingeniería, Universidad del Bío-Bío, Concepción 4051381, Chile; cristhia@ubiobio.cl (C.A.); cesar.navarrete1401@alumnos.ubiobio.cl (C.N.)
* Correspondence: cristhian.aguilera@uss.cl

Abstract: In blueberry farming, accurately assessing maturity is critical to efficient harvesting. Deep Learning solutions, which are increasingly popular in this area, often undergo evaluation through metrics like mean average precision (mAP). However, these metrics may only partially capture the actual performance of the models, especially in settings with limited resources like those in agricultural drones or robots. To address this, our study evaluates Deep Learning models, such as YOLOv7, RT-DETR, and Mask-RCNN, for detecting and classifying blueberries. We perform these evaluations on both powerful computers and embedded systems. Using Type-Influence Detector Error (TIDE) analysis, we closely examine the accuracy of these models. Our research reveals that partial occlusions commonly cause errors, and optimizing these models for embedded devices can increase their speed without losing precision. This work improves the understanding of object detection models for blueberry detection and maturity estimation.

Keywords: blueberry detection; maturity estimation; edge computing; smart agriculture; computer vision; machine learning

Citation: Aguilera, C.A.; Figueroa-Flores, C.; Aguilera, C.; Navarrete, C. Comprehensive Analysis of Model Errors in Blueberry Detection and Maturity Classification: Identifying Limitations and Proposing Future Improvements in Agricultural Monitoring. *Agriculture* **2024**, *14*, 18. https://doi.org/10.3390/agriculture14010018

Academic Editors: Maciej Zaborowicz and Jakub Frankowski

Received: 14 November 2023
Revised: 12 December 2023
Accepted: 13 December 2023
Published: 22 December 2023

1. Introduction

In modern agriculture, accurately determining the number and maturity of blueberries is essential for identifying the ideal harvest time. With noticeable variations in maturity levels among blueberry clusters [1], obtaining accurate and timely information is vital in enhancing productivity, reducing costs, and maximizing profits. This challenge has guided research efforts toward automating such assessments, offering a more data-driven and efficient strategy for determining the optimal harvesting period.

Recognizing the importance of this problem, agricultural sector researchers have achieved substantial advancements, particularly in Deep Learning applications, with convolutional neural networks (CNNs) leading these developments. Known for their exceptional ability to process complex visual data, CNNs excel in a variety of intricate tasks such as object recognition, image classification, and instance segmentation, all of which are highly valuable in numerous agricultural applications (e.g., [2–9]). These applications underscore the versatility and adaptability of CNNs in meeting the diverse challenges faced by the agricultural sector.

In blueberry detection and maturity estimation, considerable advancements have been made, leveraging a fusion of machine learning and computer vision techniques (e.g., [10,11]). Innovative approaches, including hyperspectral imaging, partial least squares regression, and Deep Learning models, have been employed to extract color and texture

features for maturity classification [12,13]. Notable contributions include pipelines designed by Gonzalez et al. [14] and Ni et al. [1], which use CNN models for classifying blueberry traits, including maturity estimation.

More novel approaches such as the work conducted by Mu et al. [15] significantly enhanced the accuracy and efficiency of blueberry quality detection, leveraging Deep Learning for classification tasks. Obsie et al. [16] demonstrated the viability of various machine learning algorithms in developing predictive models for blueberry yield prediction. MacEachern et al. [17] successfully trained models to identify wild blueberry ripeness stages, achieving high mean average precision (mAP) values for two and three types of ripeness, alongside an impressive runtime inference compared to previous approaches.

While advancements in Deep Learning for agriculture are significant, a critical area of research remains in assessing their real-world viability, especially on embedded devices like those in agricultural robots and drones. These autonomous systems require careful consideration of processing power, size, weight, and connectivity, typical of edge computing environments. Furthermore, cost-effectiveness in processing is crucial for widespread adoption. Although edge computing devices are affordable and compact, they present challenges in balancing speed and precision. Effectively applying these methods for tasks such as blueberry detection and maturity estimation in practical settings is an evolving field.

Additionally, the practical application of these advancements warrants further exploration. Most existing research involves image acquisition in controlled environments, which only partially represents the complexities of real-world conditions. This discrepancy between laboratory and field settings and the computational challenges addressed in this study highlight the need for more research to bridge the gap between theoretical models and their practical implementation in agriculture.

Contributions of the Study

This study evaluates various state-of-the-art models for detecting and estimating blueberry maturity across multiple devices, assessing their real-world application potential. Utilizing a dataset specially curated for this purpose, the performance of these models is examined, employing the Type-Influence Detector Error (TIDE) method for a detailed analysis of prevalent issues. This approach identifies critical areas needing improvement and facilitates a discussion on future research directions, potentially leading to more refined and efficient methods in agricultural technology.

The main contributions of this work are as follows:

- A new and publicly available dataset of blueberries for object detection tasks covering various maturity stages has been created, captured, and labeled. While other datasets include object detection labels, this incorporates maturity classification.
- This study offers novel insights into the performance of current models, particularly in terms of runtime and error analysis. State-of-the-art blueberry detection and maturity estimation models have been thoroughly evaluated, with the errors identified and quantified using the TIDE method. These insights are crucial for guiding future research efforts.
- An essential contribution of our study is the exploration of model adaptability in edge computing environments, explicitly examining their computational demands and performance. We show that some object detection models can operate in real-time on edge devices while maintaining their ability to detect and classify blueberry maturity effectively. Although runtime information for these models on embedded devices is known, the impact of optimization techniques required for these devices on their capability for blueberry detection and maturity estimation was previously unexplored.
- The code associated with the evaluations is made available to promote research reproducibility and encourage further advancements in this field. The code can be accessed at https://github.com/ngunsu/bb2023 (accessed on 13 November 2023).

2. Materials and Methods

2.1. Image Dataset

In this study, a set of blueberry images was collected from a plantation situated in Quillón, a town in the Ñuble Region of Chile. This region is known for its mild microclimate, exhibiting an average annual temperature of 14.9 °C. Typical January temperatures fluctuate between 27 and 30 °C, while the annual precipitation ranges from 700 to 1000 mm, with most rainfall occurring between April and September. The images were captured on three separate occasions, from late October to early December 2021, during sunny days when the temperatures exceeded 20 °C.

2.1.1. Image Acquisition

The image acquisition process employed a Nikon Coolpix B700 and a Basler acA2440-20gc camera, both firmly mounted on a SOLIGOR WT-330A tripod to ensure stability during the capture process. Figure 1 displays the configuration of this imaging system. The Nikon Coolpix B700, equipped with a 60× optical Zoom-NIKKOR glass lens (4.3–258 mm), captures images with a resolution of 5184 × 3888 pixels. Meanwhile, the Basler acA2440-20gc, using a Fujinon HF9HA-1B Lens (9 mm 1.5MP 2/3″ f/1.4 C-Mount), captures images at 2448 × 2048 pixels.

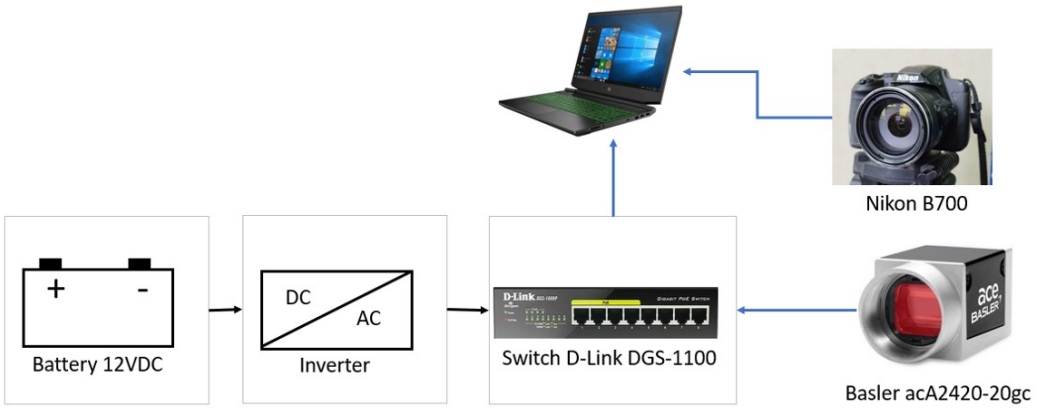


Figure 1. Camera acquisition setup showcasing two cameras: the Nikon Coolpix B700 and the Basler acA2440-20gc. The Nikon Coolpix B700 is directly connected to a notebook, while the Basler acA2440-20gc connects to the notebook via an ethernet switch. The notebook facilitates the capture process. Power to the ethernet switch, vital for the Basler camera's operation, is supplied by a 12-volt battery.

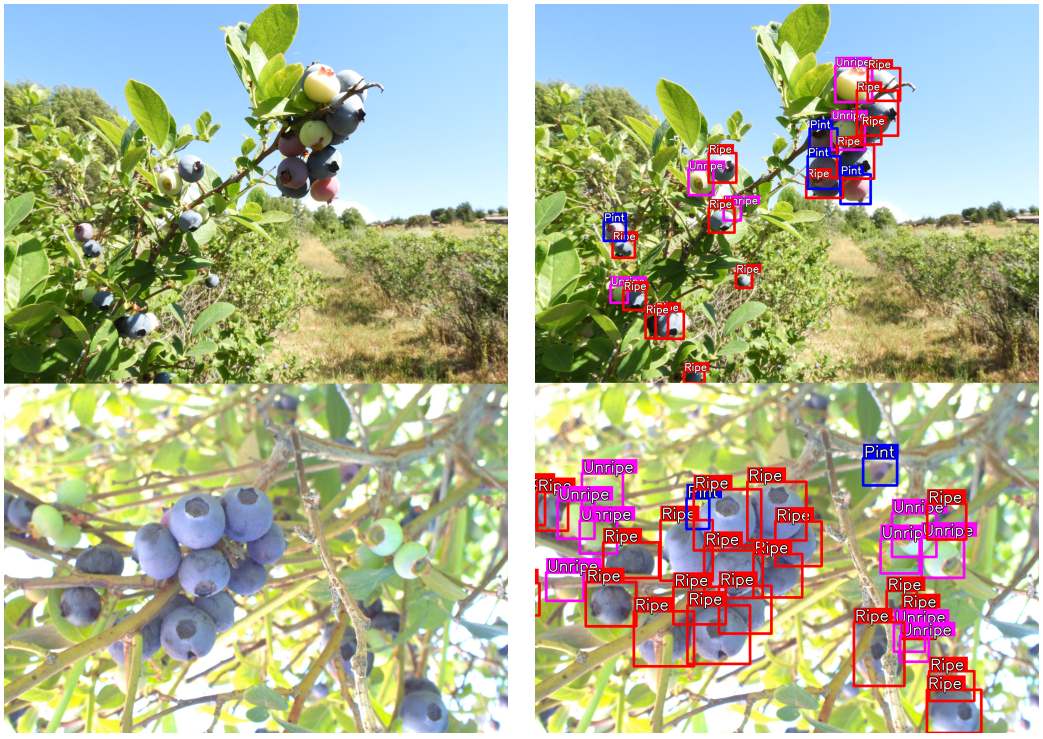
Using the previously described camera setup, approximately 500 images were captured from different locations within the plantation. The cameras' automatic illumination settings were used for each image capture, and any specialized adjustments were deliberately avoided. This approach was intentionally chosen to create challenging conditions where the variable lighting could affect the blueberries' coloration. This approach aims to generate edge-case scenarios, thereby providing a comprehensive evaluation of our models' performance under diverse and demanding environmental conditions.

2.1.2. Image Labeling

Image labeling is crucial in creating datasets for machine learning applications. In this context, labels serve as the ground truth that a machine learning model aims to learn. For our study, labeling involved meticulously outlining each blueberry with rectangular bounding boxes and categorizing them according to their maturity levels. The categories were defined as follows: berries with green or reddish tones were classified as *unripe*, those with light purple to darker red hues were *pint*, and berries showing blue or dark purple coloration were labeled as *ripe*.

Label Studio 1.5.0 was employed to label the images to facilitate this process. Three individuals participated in this detailed labeling process, carefully drawing rectangular

bounding boxes around the blueberries and assigning the appropriate ripeness category. Throughout the labeling process, several images were discarded if a labeler could not reliably assess the ripeness of any blueberries in the image. This quality control step resulted in a refined dataset of 265 images, representing a wide range of ripeness stages. Figure 2 shows examples of these annotations, highlighting the diversity of ripeness stages included in the dataset.



(**a**) Original image (**b**) Labeled image

Figure 2. Representative image from the dataset. The image on the left presents the original capture, while the image on the right displays the same capture with manually added labels illustrating blueberry locations and their respective maturity categories.

2.1.3. Dataset Splits

Of the 265 images labeled, 85% were allocated for training and 15% for testing purposes. The training data were also subdivided into two sets: a primary training set and a validation set. This subdivision followed the same 15–85% ratio, with 85% of the images used for training and 15% for validation. Each split was conducted through random partitioning to ensure variety and unpredictability in the data distribution.

Crucially, the datasets were manually reviewed to guarantee that no clusters of blueberries were duplicated across the sets, thus maintaining distinct and unique image sets for training, validation, and testing. Table 1 displays the final distribution and number of images across these sets, illustrating the breakdown of the dataset for the different phases of the machine learning process.

Table 1. Distribution of images across training, validation, and testing subsets within the image dataset.

Data	Training	Validation	Test	Total
Images	190	33	42	265

Table 2 presents a comprehensive breakdown of the generated labels for each image, based on its usage in training and evaluation, as well as its maturity level (see Figure 3). It

is important to note that the dataset exhibits a slight imbalance, with the *pint* class having fewer instances than other classes. This imbalance can be attributed to the date of the image captures.

Table 2. Classification and distribution of labels corresponding to blueberry maturity stages.

Class	Train	Validation	Test	Total
unripe	2825	539	680	4044
pint	431	66	170	667
ripe	3271	556	628	4455

(**a**) (**b**) (**c**)

Figure 3. Examples of blueberry labeling: (**a**) unripe, (**b**) pint, and (**c**) ripe.

2.2. Model Training and Evaluation

2.2.1. Model Training

Training and evaluation were conducted on three distinct object detection architectures for identifying and classifying the maturity of blueberries: YOLOv7 (including YOLOv7-tiny, YOLOv7-w6, and YOLOv7-default) [18], Mask-RCNN [19], and RT-DETR-L [20]. The primary goal of this study was to assess the impact of different models, with their varying number of parameters and runtime speeds, on the detection and classification accuracy within our dataset.

A crucial step in training these models was the hyperparameter tuning phase, where a range of adjustments was explored. Fine-tuning strategies were also implemented, using pre-trained models from the COCO dataset to enhance the models' performance. This involved rigorously evaluating various learning rates, a critical factor in how quickly a model learns during training and, thus, affects its overall performance. Furthermore, several data augmentation techniques were used to artificially increase the dataset's size. Given the small size of our training set, this approach was especially advantageous. Data augmentation, involving image transformations like rotation or vertical mirroring, allowed us to generate multiple samples from a single image, thereby improving the model's learning efficiency and generalization capability. The optimal hyperparameters for each model found through grid search are listed in Table 3.

The computational constraints of the GPU setup required minor image size modifications to ensure successful model training. All experiments were conducted on a computer system with a 12th Gen Core i7 CPU, 32GB of RAM, a 1TB SSD, and an NVIDIA RTX3080TI 10GB GPU.

Table 3. Hyperparameters utilized for training each object detector in our experimental analysis, with a uniform training duration of 350 epochs for all models. All models were pre-trained on the COCO dataset.

Model	Image Size	lr	Augmentation
YOLOv7-tiny	640 × 640	0.01	Translation, Scale, Rotation, Vertical Flip, Horizontal Flip, Copy Paste, Mosaic
YOLOv7-default	640 × 640	0.001	Translation, Scale, Rotation, Vertical Flip, Horizontal Flip, Copy Paste, Mosaic
YOLOv7-w6	1280 × 1280	0.010	Translation, Scale, Rotation, Vertical Flip, Horizontal Flip, Copy Paste, Mosaic
RT-DETR-L	640 × 640	0.001	Translation, Vertical Flip, Horizontal Flip, Mosaic
Mask-RCNN	[800, 1333] × [800, 1333]	0.010	Vertical Flip, Horizontal Flip

2.2.2. Model Evaluation

The average mean precision is a standard metric for evaluating object detection models, assessing both the accuracy of the detected objects and the model's confidence in these detections. Among its variations, mAP50 is widely used, where detection is considered accurate if the intersection-over-union (IoU) between the predicted bounding box and the ground truth is at least 50%. The calculation of mAP50 involves sorting all detections by their confidence scores, determining each detection as a true positive or a false positive based on the IoU threshold, and then calculating precision and recall at each threshold level. The final mAP50 score is an average of these precision values, taken at the points where recall changes, across all classes in the dataset.

Similarly, mAP75 follows the same calculation process but with a stricter IoU threshold of 75%, providing a more rigorous evaluation of the model's accuracy. This metric is especially relevant in our study, where precision in the localization of objects is crucial.

Additionally, the precision, recall, and F1 score of the models are evaluated. Precision measures the proportion of correct identifications made by the model, while recall measures the proportion of actual positives that were correctly identified. The F1 score is a measure that combines precision and recall, providing a balance between them. These metrics can be mathematically represented as follows:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{1}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{3}$$

These metrics offer a comprehensive evaluation of the model's performance in object detection tasks.

Regarding runtime, the same procedure is followed for each model. First, a warmup phase of 100 runs is started, which initializes the GPU. This step ensures the system is fully operational before beginning the measurements. After the warmup, an additional 100 runs are conducted, and the inference times are meticulously recorded. The runtime is then determined by calculating the average time across these runs.

2.3. Edge Computing

Edge computing emerged as a solution for situations when the acquired data must be processed on the spot without the possibility of being transmitted to a remote server. Therefore, it is a suitable solution for technology's deployment in rural areas, which usually lack Internet connectivity. Furthermore, edge computing devices' compact size and energy efficiency make them ideal for integration into drones or mobile robots, rendering them fit for real-world applications. This attribute expands the technology's accessibility, balancing computational power, energy consumption, and cost-effectiveness. For all the above considerations, we adopted edge computing as a compelling approach for real-time blueberry detection and classification.

Selecting appropriate devices for edge computing is crucial, particularly when aligned with the specific requirements of an application. This study focuses on NVIDIA's Jetson line, particularly the Jetson AGX Xavier and Jetson AGX Orin models. These models were selected for their superior technical features and software compatibility. Essentially, both Jetson devices are compact computers equipped with integrated GPUs and capable of being powered by batteries. This setup enables the execution of Deep Learning models with relatively lower costs than traditional desktop setups. Specifically, we chose the Jetson AGX Xavier and Orin models for their proficiency in efficiently running advanced AI models, such as RT-DETR and YOLOv7, which are integral to our research. This efficiency marks a significant improvement over earlier models like the Jetson Nano, which is limited by its outdated software capabilities. The technical specifications of the Jetson AGX Xavier and Orin are detailed in Table 4, underscoring their suitability for our research.

Table 4. Technical specifications of the NVIDIA Jetson devices evaluated in this study.

Device	Specifications
Jetson AGX Xavier	CPU: 8-core NVIDIA Carmel ARM v8.2 64-bit CPU @ 2.26 GHz GPU: 512-core Volta GPU with Tensor Cores DLA: 2 × NVDLA engines RAM: 16 GB 256-bit LPDDR4x@137 GB/s Storage: 32 GB eMMC 5.1 onboard Power: 9 V 20 V DC
Jetson AGX Orin	CPU: 2 × 12-core NVIDIA Arm® Carmel CPU@2.75 GHz GPU: 2 × NVIDIA Ampere architecture Tensor Cores and 2 × NVIDIA Volta architecture Tensor Cores DLA: 2 × NVDLA engines Memory: 128 GB/s 256-bit LPDDR4x | 200 GB/s 2048-bit LPDDR5 Storage: 1 × 10GbE, 1 × 5GbE, 1 × 2.5GbE, 1 × 1GbE Power: 9 V 36 V DC

It is crucial to emphasize that although the Jetson devices—namely the AGX Xavier and AGX Orin—are powerful mini-computers, they are primarily designed for inference tasks rather than for the training phase of machine learning models. Consequently, in our research, these embedded systems will be utilized exclusively for evaluating the runtime performance of machine learning models that have already been trained. The training phase of these models will be conducted on more robust desktop GPUs.

2.4. Optimizing Model Runtime

Deploying Deep Learning models on embedded devices often necessitates a post-processing phase to optimize them for efficient runtime performance. Typically, these models are designed for desktop-grade GPUs, and their performance on embedded devices is comparatively lower due to the limitations of these devices' internal GPUs, such as reduced memory capacity and fewer GPU cores. Therefore, it is necessary to adapt the models for these devices to enhance their runtime speed once they are trained. Several techniques are employed for this purpose. For instance, knowledge distillation, as described

in [21], involves training a compact model to emulate the behavior of a larger, more complex model, making the smaller version more suitable for embedded devices. Another prevalent technique is quantization, which accelerates network inference by utilizing lower precision computations, like 16-bit, 8-bit, or even 1-bit precision models [22].

However, this study adopts a more direct optimization approach using TensorRT 8.5.0.2 [23], a software tool developed by NVIDIA. TensorRT effectively reduces the model size and enhances runtime performance through quantization, converting 32-bit floating-point computations to 16-bit or 8-bit formats. This adaptation increases the runtime speed as GPUs process these calculations more quickly. TensorRT also implements layer and tensor fusion, combining operations to run faster and fully utilizing GPU capabilities. TensorRT takes a trained model as the input and produces a new, more efficient version for inference. This type of optimization is essential for applications that require high-speed processing.

While TensorRT offers substantial benefits, assessing its impact on the network's performance is crucial. The trade-off between runtime efficiency and model accuracy is a significant factor in this assessment. Consequently, our article focuses on an in-depth analysis of TensorRT's optimization effects, particularly in real-time edge detection and maturity estimation of blueberries on edge devices.

2.5. The Type-Influence Detector Error

The Type-Influence Detector Error (TIDE) [24] analysis is a tool to examine the types of errors made by an object detector and how these errors affect the mean average precision metric (mAP). It provides a detailed perspective on specific categories of errors and their contribution to the detector's overall performance. Essentially, TIDE quantifies the impact of each error type on the total mAP (denoted as dAP), offering an estimate of potential mAP improvement if a particular error was effectively addressed.

TIDE analysis gives information about the following:

1. Misclassification errors (Cls), which occur when the detected object is incorrectly classified;
2. Localization errors (Loc), which arise when the algorithm accurately classifies an object but inaccurately localizes it, underscoring the need for enhancements in object detection algorithms;
3. Combined misclassification and mislocalization errors (Both);
4. Duplication errors (Dup), which occur when an object is detected multiple times;
5. Background errors (Bkg), which occur when the algorithm wrongly identifies parts of the background as objects;
6. Missed errors (Miss), which represent overlooked ground truth tags by the algorithm;
7. False positive (FP) errors, depicting instances where the algorithm mistakenly identifies non-objects as objects;
8. False negative (FN) errors result when the algorithm fails to detect an existing object.

3. Results

3.1. Blueberry Detection and Maturity Estimation

Building on the findings in [17], the empirical evaluation has been broadened to encompass a range of more recent and diverse models. Table 5 presents the results of our trained models for blueberry detection and maturity estimation. This analysis adheres to the methodology described in [17], particularly employing a stringent mean average precision (mAP) criterion of IOU 75%. The findings highlight that the mAP scores for most models range between 0.3 and 0.5, suggesting a moderate accuracy level. Among the models, MASK-RCNN stands out for its superior accuracy, though it is also the slowest in runtime. Additionally, the analysis reveals a consistent precision level across all maturity levels, indicating that no single class disproportionately contributes to errors despite the imbalance in the dataset.

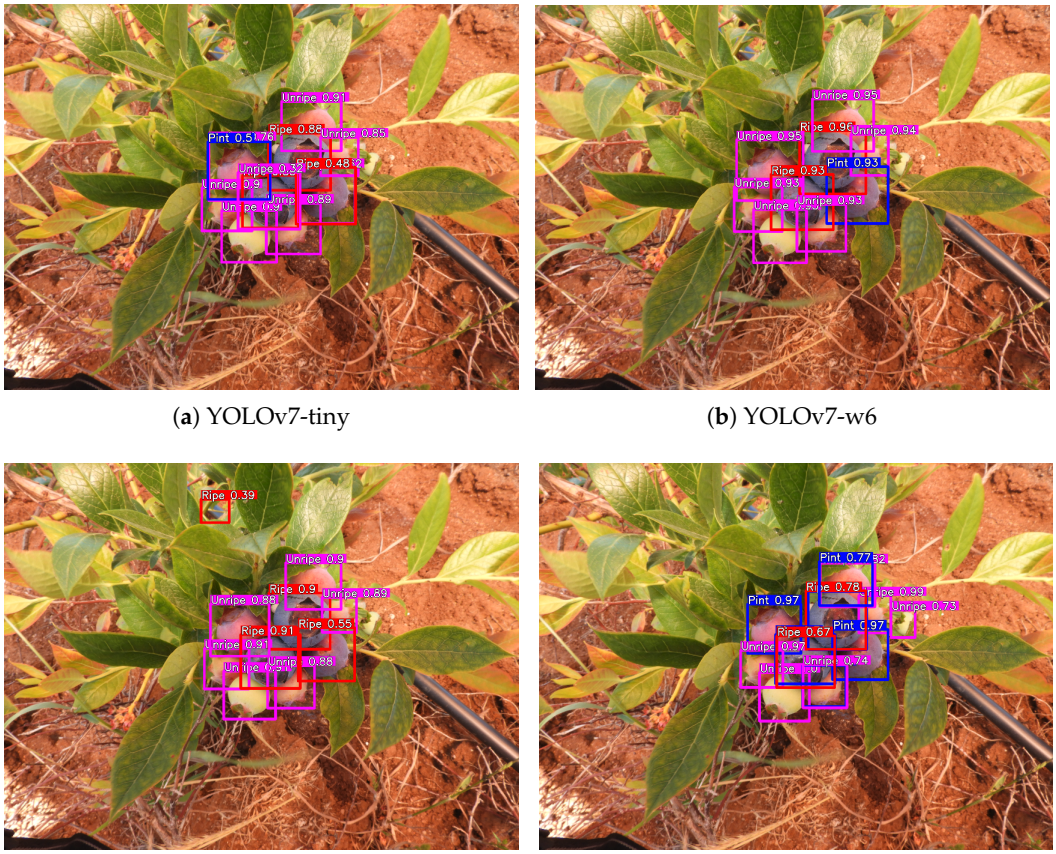
Table 5. This table offers a comparative assessment of various YOLOv7 configurations, RT-DETR-L, and Mask-RCNN in detecting blueberries at three stages of maturity—ripe, pint, and unripe. The performance metrics are evaluated on an NVIDIA RTX3080TI GPU, using mAP75.

Model	Class	Precision	Recall	F1	mAP75	Runtime (ms)
YOLOv7-tiny	Ripe	0.547	0.387	0.443	0.330	3.308
	Pint	0.568	0.433	0.489	0.364	
	Unripe	0.485	0.323	0.388	0.231	
	All	0.533	0.380	0.443	0.309	
YOLOv7-default	Ripe	0.626	0.456	0.528	0.435	8.059
	Pint	0.641	0.508	0.567	0.432	
	Unripe	0.605	0.415	0.492	0.348	
	All	0.624	0.460	0.530	0.405	
YOLOv7-w6	Ripe	0.598	0.500	0.544	0.445	19.551
	Pint	0.631	0.494	0.554	0.431	
	Unripe	0.591	0.457	0.516	0.381	
	All	0.607	0.484	0.539	0.419	
RT-DETR-L	Ripe	0.547	0.454	0.496	0.416	11.551
	Pint	0.575	0.348	0.434	0.314	
	Unripe	0.518	0.396	0.449	0.309	
	All	0.547	0.399	0.462	0.346	
Mask-RCNN	Ripe	0.612	0.490	0.543	0.447	34.301
	Pint	0.680	0.574	0.622	0.558	
	Unripe	0.582	0.488	0.530	0.426	
	All	0.625	0.518	0.565	0.477	

Figure 4 presents the detection results from four distinct models applied to the same image. This comparative analysis reveals that across this particular sample, the detection bounding boxes generated by each model are similar. Notably, the results from Mask-RCNN align more closely with the actual contours of the blueberries. Furthermore, the RT-DETR model uniquely identifies one blueberry that the other models overlooked. Of particular interest is that Mask-RCNN is the only detector that accurately identifies the pint berries in this sample, demonstrating its superior precision in distinguishing between different maturity stages of the blueberries.

3.2. The TIDE Analysis

Table 6 presents our TIDE analysis, which is based on the results from the previous subsection, explicitly targeting mAP75. This table highlights that the predominant error is the models' inability to detect all blueberry instances, leading to many false negatives. This issue underscores the need for the enhanced localization of blueberries, especially those partially obscured by plant foliage or too small for the network to detect accurately. Additionally, localization error (Loc) is the second primary source of inaccuracies. This can be attributed to the natural clustering of blueberries, where, often, a single detection may encompass parts of adjacent blueberries, leading to skewed bounding boxes. Furthermore, there is room for improvement in maturity classification. The analysis suggests that an average improvement of over five percent is achievable with more accurate classification. Also, after performing a qualitative analysis of the detection results on the test set, we discovered that most classification errors occur in two scenarios: first, when a blueberry transitions between stages, such as partially ripe and unripe, and second, when the bounding box inadvertently includes background elements like leaves, affecting the color analysis. Figure 5 illustrates some of these common errors made by the object detectors.



(**a**) YOLOv7-tiny (**b**) YOLOv7-w6

(**c**) RT-DETR-L (**d**) Mask-RCNN

Figure 4. Qualitative comparison of detection results using YOLOv7-tiny, YOLOv7-w6, RT-DETR-L, and Mask-RCNN: this image presents a side-by-side visualization of the detection results from each model in identifying the maturity stages of blueberries. The color coding for the maturity stages is as follows: blue indicates pint blueberries, red represents ripe blueberries, and pink denotes unripe blueberries.

(**a**) Cls error (**b**) Loc error (**c**) Miss error

Figure 5. Examples of typical errors encountered by object detectors. Figure (**a**) shows two pint blueberries incorrectly identified as unripe. In Figure (**b**), the bounding box has a low intersection-over-union ratio, encompassing only a portion of a blueberry. Figure (**c**) illustrates a missed detection where the object detector fails to recognize a blueberry hidden among the plant's leaves.

Table 6. Comparative analysis of types of object detection errors for three classes as identified by TIDE. Each value indicates the contribution of a specific error type to the overall mAP.

Type	Cls	Loc	Both	Dup	Bkg	Miss	FP	FN
YOLOv7-tiny	5.71	32.65	0.30	0	0.27	10.66	10.35	43.30
YOLOv7-default	5.84	29.46	0.31	0	0.35	10.18	10.64	38.33
YOLOv7-w6	6.32	26.78	0.26	0	0.26	11.39	9.45	39.15
RT-DETR-L	6.44	38.24	0.33	0	0.54	6.63	11.34	41.08
Mask-RCNN	2.64	27.88	0.14	0	0.77	10.22	9.31	35.69

3.3. Edge Computing

As previously mentioned, Deep Learning models optimized for real-time processing on high-end GPUs often struggle to perform under similar conditions on edge devices, necessitating post-processing optimization. In this study, TensorRT was applied to the models evaluated in earlier sections, assessing their performance using 16-bit precision. The results of this optimization are presented in Figure 6. The figure shows that most methods could not achieve real-time performance on the Jetson AGX Xavier, an affordable embedded vision system. However, on the more expensive and high-end Jetson AGX, not only YOLOv7-tiny but also YOLOv7-default could be run in real-time. Networks like Mask-RCNN, despite their accuracy, proved unsuitable for real-time tasks. In this context, YOLOv7-default emerges as a balanced choice, effectively bridging the gap between accuracy and runtime performance.

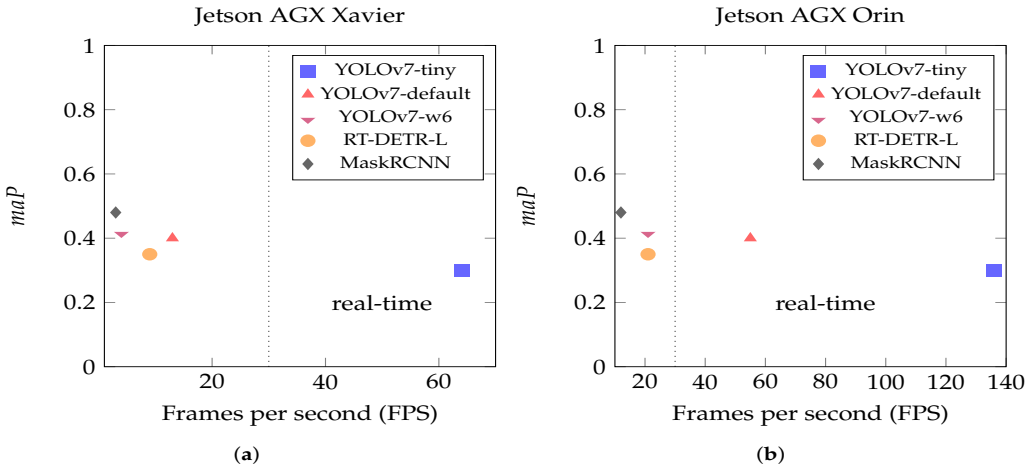


Figure 6. Comparative runtime results of the object detection models evaluated on Jetson AGX Xavier (**a**) and Jetson AGX Orin (**b**), following optimizations discussed in Section 2.3.

Regarding mAP, it is noteworthy that the optimization process did not significantly alter the mAP for any of the models. Only minor and relatively insignificant improvements were observed across most models, possibly due to the noise reduction during the optimization. The Mask-RCNN model exhibited the most noticeable change post-optimization, particularly when tailored for Jetson cards, though this change was also minimal. Table 7 details the variations in mAP for a selected model across different devices.

In conclusion, the results indicate that runtime optimization does not significantly alter the performance of the models in the task of detecting and estimating the maturity of blueberries, thus facilitating their deployment on embedded devices. However, it is noteworthy that only a select few models are capable of real-time operation, suggesting that further optimization may be necessary. This is particularly relevant as more modern, transformer-based models begin to gain prominence in this field.

Table 7. Comparison of model performance optimized with TensorRT on various devices: this table displays the mean average precision (mAP) at a threshold of 75 for different models, illustrating the impact of TensorRT optimization across multiple devices.

Model	RTX 3080TI 16bit	Jetson Orin 16bit	Jetson Xavier 16bit
YOLOv7-tiny	0.331	0.329	0.329
YOLOv7	0.433	0.434	0.435
YOLOv7-w6	0.436	0.419	0.421
RT-DETR-L	0.321	0.321	0.319
Mask-RCNN	0.462	0.462	0.460

3.4. Discussion

Detecting and estimating the maturity of blueberries remains a complex task, primarily due to background elements like leaves and natural occlusions inherent to the plant. Most of the research in this area, including our study, depends on static images from a single viewpoint, which may limit the accuracy potential. From our findings, we hypothesize that in field applications, capturing multiple images of the same blueberry cluster from various viewpoints could significantly enhance the detection process. Selecting images from multiple angles could reduce the number of occluded blueberries, and analyzing clusters from these different perspectives might provide critical supplementary information. This approach could improve detection algorithm performance by offering a more detailed view of each cluster. However, this hypothesis requires further exploration, as it involves challenges such as rapid processing speeds and sophisticated tracking capabilities, which have not been extensively investigated in blueberry research.

In this context, creating datasets that more closely mirror real-world conditions is essential for advancing research in this field. Shifting our focus from static images to video data is particularly important, as it aligns more directly with the practical needs of the industry. This change will allow future research to address the challenges in agricultural settings more effectively. The limitations highlighted in existing datasets, including the one used in our study, emphasize the urgency of this transition.

Finally, enhancing the performance of detection models on embedded devices presents distinct challenges. Our findings reveal that not all embedded systems can run advanced detection techniques in real-time. Furthermore, object detection and maturity estimation are often just part of a more extensive system, especially in robotic applications. This means the runtime must accommodate additional computations for functionalities such as tracking and navigation. Consequently, there is a critical need for ongoing advancements in these technologies aimed at boosting performance across a range of devices, including more cost-effective options. Future research should focus on optimizing processing efficiency to encourage broader adoption and practical implementation of these technologies. Such advancements could make these solutions more widely available and cost-effective, potentially transforming agricultural practices on farms of every size.

4. Conclusions

This study has conducted an extensive analysis of various advanced Deep Learning techniques for detecting and estimating the maturity of blueberries. Our investigations reveal that while current models are good at localizing individual blueberries, they face challenges from the inherent constraints of the detection techniques and from the natural characteristics of blueberry plants, where berries often remain partially occluded.

A significant observation from our research is the difficulty in achieving accurate localization due to the clustered nature of blueberries and the viewpoint from where the image was captured. These complexities often result in detection inaccuracies, such as misclassification or imprecise bounding boxes, as highlighted by our TIDE analysis, which indicates a significant prevalence of false negatives and localization errors.

Regarding edge computing, our experiments show that some models can perform in real-time on edge devices without significantly losing precision. However, the efficiency of

these models varies, with some, like Mask-RCNN, exhibiting higher accuracy but longer runtime, which restricts their real-time application. In contrast, models such as YOLOv7-default strike a more effective balance between accuracy and processing speed, making them more suitable for real-time tasks.

Finally, our findings provide valuable insights into the capabilities and limitations of current techniques in blueberry detection and maturity estimation. They emphasize the complexity of this task, influenced by both the nature of the blueberry plants and the limitations of existing detection models.

Author Contributions: Conceptualization, C.A.A., C.F.-F. and C.A.; formal analysis, C.A.A. and C.F.-F.; funding acquisition, C.A.; investigation, C.A.A., C.F.-F., C.A. and C.N.; methodology, C.A.A. and C.F.-F.; software, C.A.A. and C.N.; supervision, C.A.; writing—original draft, C.A.A., C.F.-F. and C.A.; writing—review and editing, C.A.A., C.F.-F. and C.A. All authors have read and agreed to the published version of this manuscript.

Funding: This research was funded by the National Research and Development Agency through the FONDEF project ID21I10256 and Project INES I+D 22-14 Bio-Bio University.

Data Availability Statement: Data are contained within the article.

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

References

1. Ni, X.; Li, C.; Jiang, H.; Takeda, F. Deep learning image segmentation and extraction of blueberry fruit traits associated with harvestability and yield. *Hortic. Res.* **2020**, *7*, 110. [CrossRef] [PubMed]
2. Kamilaris, A.; Prenafeta-Boldú, F.X. Deep learning in agriculture: A survey. *Comput. Electron. Agric.* **2018**, *147*, 70–90. [CrossRef]
3. Gunawan, G.; Zarlis, M.; Sihombing, P.; Wage, S. Optimization of the CNN model for smart agriculture. *IOP Conf. Ser. Mater. Sci. Eng.* **2021**, *1088*, 012029. [CrossRef]
4. Jin, Y. Detection of Crop Leaf Diseases and Insect Pests Based on Improved Faster R-CNN. *Fresenius Environ. Bull.* **2021**, *30*, 7278–7290.
5. Mondal, S.; Banerjee, S.; Mukherjee, S.; Sengupta, D. Plant Disease Detection Using Ensembled CNN Framework. *Comput. Sci.-AGH* **2022**, *23*, 323–335. [CrossRef]
6. Manuel Lopez-Correa, J.; Moreno, H.; Ribeiro, A.; Andujar, D. Intelligent Weed Management Based on Object Detection Neural Networks in Tomato Crops. *Agronomy* **2022**, *12*, 2953. [CrossRef]
7. Zhang, X.; Cui, J.; Liu, H.; Han, Y.; Ai, H.; Dong, C.; Zhang, J.; Chu, Y. Weed Identification in Soybean Seedling Stage Based on Optimized Faster R-CNN Algorithm. *Agriculture* **2023**, *13*, 175. [CrossRef]
8. Qiao, M.; He, X.; Cheng, X.; Li, P.; Luo, H.; Tian, Z.; Guo, H. Exploiting Hierarchical Features for Crop Yield Prediction Based on 3-D Convolutional Neural Networks and Multikernel Gaussian Process. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2021**, *14*, 4476–4489. [CrossRef]
9. Ju, S.; Lim, H.; Ma, J.W.; Kim, S.; Lee, K.; Zhao, S.; Heo, J. Optimal county-level crop yield prediction using MODIS-based variables and weather data: A comparative study on machine learning models. *Agric. For. Meteorol.* **2021**, *307*, 108530. [CrossRef]
10. Tellaeche, A.; BurgosArtizzu, X.P.; Pajares, G.; Ribeiro, A.; Fernández-Quintanilla, C. A new vision-based approach to differential spraying in precision agriculture. *Comput. Electron. Agric.* **2008**, *60*, 144–155. [CrossRef]
11. Behera, S.K.; Rath, A.K.; Sethy, P.K. Maturity status classification of papaya fruits based on machine learning and transfer learning approach. *Inf. Process. Agric.* **2021**, *8*, 244–250. [CrossRef]
12. Yang, C.; Lee, W.S.; Gader, P. Hyperspectral band selection for detecting different blueberry fruit maturity stages. *Comput. Electron. Agric.* **2014**, *109*, 23–31. [CrossRef]
13. Tan, K.; Lee, W.S.; Gan, H.; Wang, S. Recognising blueberry fruit of different maturity using histogram oriented gradients and colour features in outdoor scenes. *Biosyst. Eng.* **2018**, *176*, 59–72. [CrossRef]
14. Gonzalez, S.; Arellano, C.; Tapia Farias, J. DeepBlueBerry: Quantification of Blueberries in the Wild Using Instance Segmentation. *IEEE Access* **2019**, *7*, 105776–105788. [CrossRef]
15. Mu, C.; Yuan, Z.; Ouyang, X.; Sun, P.; Wang, B. Non-destructive detection of blueberry skin pigments and intrinsic fruit qualities based on deep learning. *J. Sci. Food Agric.* **2021**, *101*, 3165–3175. [CrossRef] [PubMed]
16. Obsie, E.Y.; Qu, H.; Drummond, F. Wild blueberry yield prediction using a combination of computer simulation and machine learning algorithms. *Comput. Electron. Agric.* **2020**, *178*, 105778. [CrossRef]
17. MacEachern, C.B.; Esau, T.J.; Schumann, A.W.; Hennessy, P.J.; Zaman, Q.U. Detection of fruit maturity stage and yield estimation in wild blueberry using deep learning convolutional neural networks. *Smart Agric. Technol.* **2023**, *3*, 100099. [CrossRef]
18. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv* **2022**, arXiv:2207.02696.

19. Qiao, Y.; Truman, M.; Sukkarieh, S. Cattle segmentation and contour extraction based on Mask R-CNN for precision livestock farming. *Comput. Electron. Agric.* **2019**, *165*, 104958. [CrossRef]
20. Lv, W.; Xu, S.; Zhao, Y.; Wang, G.; Wei, J.; Cui, C.; Du, Y.; Dang, Q.; Liu, Y. DETRs Beat YOLOs on Real-time Object Detection. *arXiv* **2023**, arXiv:2304.08069.
21. Hinton, G.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network. *arXiv* **2015**, arXiv:1503.02531.
22. Aguilera, C.A. SBIN: A stereo disparity estimation network using binary convolutions. *IEEE Lat. Am. Trans.* **2022**, *20*, 693–699. [CrossRef]
23. Jeong, E.; Kim, J.; Ha, S. TensorRT-Based Framework and Optimization Methodology for Deep Learning Inference on Jetson Boards. *ACM Trans. Embed. Comput. Syst.* **2022**, *21*, 1–26. [CrossRef]
24. Bolya, D.; Foley, S.; Hays, J.; Hoffman, J. TIDE: A General Toolbox for Identifying Object Detection Errors. In Proceedings of the Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020.