Automatic Classification of Tree Tomato Ripeness Using Convolutional Neural Networks

David Mejia[1], Kathlen Menendez[2], Mateo Ordóñez[3], and Santiago Tugulinago[4]

Department of Mechatronics, Universidad Internacional del Ecuador, Av. Simon Bolivar 170411, Quito, Ecuador
kamenendezme@uide.edu.ec, damejiato@uide.edu.ec, wetugulinagole@uide.edu.ec, maordonezc@uide.edu.ec

Abstract. This project develops a computer-vision system for the recognition of three maturity stages of tree tomato (Solanum betaceum): green, intermediate, and ripe. The image dataset is collected in greenhouses and plantations located in the Ecuadorian highlands, considering real variations in illumination, background, viewing angle, and environmental conditions typical of these agricultural environments.
The dataset is entirely locally generated, with photographs captured in authentic production scenarios, including differences in lighting intensity, cultivation settings, capture devices, and the presence of natural surface imperfections. This approach aims to minimize dataset bias and enhance the model's ability to generalize to real-world situations in the tree-tomato production chain.
The main objective is to train a model based on convolutional neural networks (CNNs) capable of automatically classifying the fruit's maturity stage, supporting agricultural decision-making, reducing postharvest losses, and optimizing sorting and commercialization processes. Additionally, the system contributes to the advancement of smart agriculture in the Ecuadorian Andes and supports the digitalization of understudied Andean crops.

Keywords: computer vision,fruit ripeness,freshness,native species,classification

1 Introduction

The tree tomato (*Solanum betaceum*) is a characteristic fruit of the Ecuadorian highlands, whose ripening process shows strong variations in color, texture, and morphology, influenced by Andean microclimates and local agricultural practices. These factors complicate visual assessment and lead producers to rely on subjective judgment, often resulting in premature harvesting and postharvest losses.

Computer vision and deep learning have demonstrated high accuracy—often above 98%—in classifying ripeness in fruits such as bananas, mangoes, citrus fruits, and tomatoes using architectures like YOLOv8, MobileNetV2, VGG19, and optimized CNN variants [1, 2, 4, 9, 12, 17]. However, the literature reveals a lack of systems designed for native Andean fruits, whose chromatic variability

limits the direct transfer of pretrained models. This gap is reinforced by the absence of public datasets for these species [1, 2], and by the known performance drop of CNNs under small datasets or uncontrolled illumination [2, 4].

To address this gap, this project develops a CNN-based system to classify three ripeness stages of tree tomato—green, intermediate, and ripe—using images captured in real agricultural environments, following methodologies applied to bananas and other tropical fruits [1, 3]. This work supports automation, reduces postharvest losses, and contributes to agricultural digitalization in Ecuador while aligning with SDG 9 and 12.

Despite advances in fruit-ripeness detection, current models remain unsuitable for the high variability and environmental conditions of Andean crops. The lack of field-validated models and publicly available datasets highlights the need for a robust, context-specific solution capable of performing reliable classification under real-world conditions.

2 State of the Art

Scientific literature on computer vision for fruit ripeness classification has grown significantly over the past decade. Although most studies focus on export-oriented fruits such as apples, mangoes, citrus fruits, tomatoes, bananas, or kiwifruit, the methods, algorithms, and theoretical frameworks developed provide a solid foundation for addressing ripeness in tree tomatoes. The following discussion presents a structured analysis organized around the main research trends.

2.1 Classical methods based on color, texture, and morphology

Early approaches for ripeness assessment focused on models relying solely on simple visual features such as color, texture, and shape. Fracarolli [5] compiled the most widely used traditional methods, highlighting techniques based on RGB and HSV color spaces combined with morphological metrics such as area, contour, and shape descriptors. These approaches are attractive due to their low computational cost and ease of implementation; however, they are highly sensitive to variations in lighting, shadows, and complex backgrounds.

Tapia-Méndez et al. [6] integrated texture descriptors (such as LBP and GLCM) with color information, achieving accuracies close to 96% even with moderately sized datasets. Nonetheless, they reported decreased generalization when fruits exhibited natural imperfections, dirt, or bruising—conditions commonly observed in artisanal or Andean crop production.

The XAI-FruitNet model [7] introduced interpretability through heatmap visualization, highlighting the regions of the fruit used for classification. This transparency is essential for agricultural technologies intended for end-users who require validation of model decisions.

Although these classical approaches are insufficient to fully capture the high variability of tree tomatoes, they provide valuable insights into chromatic transitions essential for ripeness evaluation.

2.2 Advances in CNNs for ripeness classification

With the rapid progress of deep learning, convolutional neural networks (CNNs) have become the standard for fruit classification tasks. Yadav [9] conducted a comparative review of architectures including VGG, ResNet, and MobileNet, noting that classification performance improves when datasets exceed 1,000 images per class, reducing overfitting and increasing generalization capability.

Other studies have applied CNNs specifically to tropical fruits. In mangoes and bananas, architectures such as MobileNetV2, VGG19, and ResNet have achieved between 95% and 98% accuracy [1, 2, 4, 12]. However, Yadav et al. observed significant performance degradation under variable lighting or when datasets are limited in size [4], a recurring issue in native fruits lacking standardized databases.

Tree tomatoes pose similar challenges: abrupt color changes between ripeness stages, variable skin gloss, and notable differences between cultivars. Therefore, CNNs remain suitable but require a robust and context-specific dataset.

2.3 Phenological detection, YOLO, and field-based vision

Recent work has shifted from laboratory setups to real field environments, where ripeness must be evaluated under natural conditions. Models such as *Lightweight Models for Orchard Detection* [10] and *AppleGrowthVision* [11] demonstrated the feasibility of combining stereoscopic vision with CNNs for 3D reconstruction of phenological features.

For tropical fruits such as bananas, Martínez-Mora et al. [12] developed a dual CNN architecture achieving over 98% accuracy while simultaneously classifying ripeness and quality. This approach is relevant to tree tomatoes due to their phenotypic similarities in color and texture variability.

Additionally, attention-based systems such as *Hybrid Attention Transformer + YOLOv8* [13] have improved maturity detection directly in the field, increasing mAP and recall metrics. These advances open the door to future automatic ripeness detection in tree tomatoes while still on the plant.

However, such solutions require extensive annotated datasets and more capable hardware, which may limit adoption by small-scale Ecuadorian farmers.

2.4 Hybrid and multitask approaches

A growing trend combines CNNs with complementary models to improve prediction performance when datasets are limited. *SmartRipen* [14], for example, fuses CNNs with GRU networks and feature selection via XGBoost, yielding significant gains in F1-score.

Meanwhile, *SwishFormer* [15] incorporates visual–tactile imagery to enhance robustness under varying brightness. These studies show that hybrid strategies can outperform traditional CNNs when images are scarce—an issue highly relevant to tree tomatoes.

In another area, *Chemical Ripening Detection* [16] used deep learning to identify chemical patterns associated with artificial ripening. Although not focused on tree tomatoes, it demonstrates the potential of AI to detect not only physiological maturity but also postharvest manipulation.

2.5 Mobile and real-time systems

For field applications, studies such as *Real-Time Freshness* [17] have shown that quantized models like MobileNetV2 and YOLOv8-nano can run on mobile devices with inference times below 0.2 seconds and accuracies above 90%. These developments make mobile applications feasible and useful for farmers and sellers.

Nevertheless, these models remain sensitive to lighting variations and require periodic calibration [17], reinforcing the need for a dedicated dataset representative of Ecuadorian agricultural environments.

3 Materials and Methods

3.1 Overall System Design

The proposed system aims to automatically classify tree tomatoes into three ripeness stages—green, intermediate, and ripe—using computer vision and deep learning techniques. RGB images were captured under real cultivation conditions and subsequently labeled using the Roboflow platform. The methodological approach is based on the comparison of three modern detection and classification architectures: YOLOv8n, YOLOv11n, and RT-DETR (a transformer-based model). These architectures were selected due to their efficiency, accuracy, and a suitable trade-off between inference speed and performance, which is crucial for future deployment in real-time agricultural monitoring systems. The complete solution integrates dataset preparation, model development, 10-fold cross-validation, and final evaluation on a fixed test set.

3.2 Proposed Solution

The proposed methodology for automatic ripeness classification of tree tomatoes is structured as a sequence of well-defined stages that enable image processing, data preparation, and training of the selected detection models. The methodological workflow encompasses the acquisition and annotation of images, followed by the training and evaluation of three state-of-the-art computer vision architectures: YOLOv8n, YOLOv11n, and RT-DETR.

The selection of YOLOv8n, YOLOv11n, and RT-DETR was driven by both technical performance considerations and practical deployment requirements for precision agriculture applications in Ecuador.

YOLO-based detectors were selected due to their single-stage detection paradigm, which enables real-time inference while maintaining high detection accuracy. According to recent studies in agricultural computer vision, YOLO architectures

consistently achieve a favorable balance between precision, recall, and computational efficiency when applied to fruit detection and maturity classification tasks.

YOLOv8n and YOLOv11n, in particular, are lightweight variants optimized for fast inference and reduced memory footprint. These models contain approximately 3–5 million parameters, making them suitable for deployment on resource-constrained hardware, such as embedded devices, edge computing platforms, and mobile systems. This characteristic is crucial for agricultural environments, where cloud connectivity may be limited and on-site processing is often required.

Furthermore, the modular design of YOLO architectures allows seamless model export to mobile and edge platforms (e.g., TensorRT, ONNX, CoreML), facilitating their integration into mobile applications, drones, and smart farming systems. Their reduced model size enables scalability across multiple agricultural zones, supporting large-scale monitoring without excessive computational costs.

RT-DETR was selected to complement YOLO-based models by introducing a transformer-based detection approach. Unlike convolution-only architectures, RT-DETR leverages global self-attention mechanisms, which allow the model to capture long-range dependencies and contextual relationships within the scene. This property is particularly beneficial in agricultural imagery, where overlapping fruits, occlusions, and complex backgrounds are common.

According to recent transformer-based detection studies, RT-DETR achieves competitive detection accuracy while maintaining real-time performance through optimized decoder structures and hybrid CNN-transformer backbones. Although RT-DETR requires higher computational resources compared to YOLOv8n and YOLOv11n, its ability to model global context provides complementary strengths, making it valuable for comparative analysis.

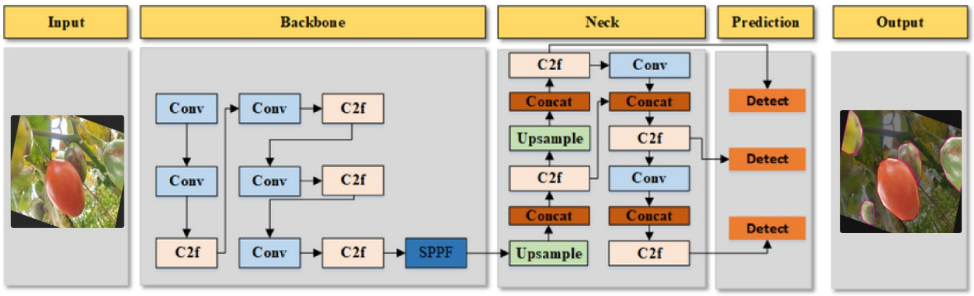

Fig. 1. Flowchart of the proposed solution.

The flowchart represents the complete detection process executed by the YOLO architecture, from the image input to the final output with the detected

objects. The process begins with the input image, which is sent to the Backbone, responsible for extracting essential features through convolution operations and C2f blocks. These blocks allow the model to capture relevant information regarding edges, textures, and tomato structures at multiple scales.

Subsequently, the extracted features are passed to the Neck, where upsampling, concatenation, and multi-scale feature fusion take place. This module enhances the representation by combining shallow and deep features, enabling robust detections of objects of different sizes and visual variations.

Next, the fused information is sent to the Prediction stage, where multiple feature levels are processed to generate the model outputs. At this stage, class probabilities for the categories ("Ripe", "Intermediate", "Green") are calculated along with their corresponding bounding boxes.

Finally, the model output displays the detections directly on the original image, where each identified fruit is highlighted with contours or bounding boxes, completing the inference process.

3.3 Computational Resources

The development, training, and evaluation of the models were carried out using a combination of local resources and cloud services. For dataset organization, environment setup, and project configuration, an ASUS ROG Strix G16 laptop was used, equipped with an Intel Core i7-13650HX processor, 16 GB of RAM, and an NVIDIA GeForce RTX 4060 Laptop GPU with 8 GB of VRAM. This configuration enabled efficient data preprocessing and model preparation.

Model training was performed using Google Colab, running in a Python 3 environment with access to an NVIDIA Tesla T4 GPU (16 GB VRAM) and extended RAM, leveraging hardware acceleration to significantly reduce computation times.

3.4 Programming Language and Libraries

The methodology was implemented using the Python programming language, due to its wide adoption in computer vision applications, availability of specialized libraries, and seamless integration with modern deep learning frameworks. The main execution environment was Google Colab, which provides native compatibility with Python 3, GPU acceleration, and a preconfigured ecosystem for modeling and analysis tasks.

The following libraries were used for model development, training, and evaluation:

- **Ultralytics:** used for training the YOLOv8n, YOLOv11n, and the transformer-based RT-DETR models. This library provides a simplified interface to execute detection, classification, and validation processes, including the automatic generation of metrics and evaluation artifacts.
- **Roboflow:** employed to manage the dataset, automate the download of images in YOLOv8 format, and maintain consistency in data structure.

- **scikit-learn:** used to implement the 10-fold Cross Validation procedure, ensuring a robust evaluation of model performance.
- **PyYAML:** utilized for reading and dynamically modifying the `data.yaml` files, allowing adjustments to dataset paths and configurations.
- **Pandas:** used for handling and analyzing the metric tables generated during training.
- **Matplotlib:** used for plotting learning curves, including the evolution of mAP and loss values over training epochs.

3.5 Evaluation Metrics

To assess the performance of the three selected models (YOLOv8n, YOLOv11n, and RT-DETR), standard metrics for object detection and classification were employed. These metrics allow measuring both the model's ability to correctly detect objects and the accuracy of its predictions.

Precision (P) Measures the proportion of correct predictions among all positive predictions made by the model.

$$Precision = \frac{TP}{TP + FP}$$

Where:

- **TP** = True Positives
- **FP** = False Positives

A high precision value indicates that the model makes few mistakes when identifying instances of a class.

Recall (R) Indicates the proportion of actual instances correctly detected by the model.

$$Recall = \frac{TP}{TP + FN}$$

Where:

- **FN** = False Negatives

A high recall value implies that the model misses few real instances.

Mean Average Precision (mAP@50) Represents the average area under the Precision–Recall curve for a fixed IoU threshold of 0.5.

$$mAP_{50} = \frac{1}{N} \sum_{i=1}^{N} AP_i(IoU = 0.50)$$

This metric evaluates how well the predicted bounding boxes match the ground truth.

Mean Average Precision (mAP@50-95) This is the most comprehensive COCO metric and the primary reference used in this project. It computes the averaged mAP over 10 IoU thresholds: 0.50, 0.55, 0.60, ..., 0.95.

$$mAP_{50:95} = \frac{1}{10N} \sum_{t=0.50}^{0.95} \sum_{i=1}^{N} AP_i(IoU = t)$$

It simultaneously evaluates precision, recall, and geometric quality of detections. Models with higher mAP@50-95 exhibit better general performance.

3.6 Hyperparameters

Table 1. Hyperparameters used in the training methodology.

Parameter	**Description**	**Value**
Image size	Input image resolution used during training and inference	640×640 px
Batch size (YOLOv8n / YOLOv11n)	Number of images processed per iteration for CNN-based detectors	16
Batch size (RT-DETR)	Reduced batch size due to higher memory consumption of transformer-based architecture	4
Epochs	Number of complete passes over the training dataset	20
Optimizer	Optimization algorithm used to update network weights	AdamW
Initial learning rate	Initial learning rate applied at the start of training	0.001
K-Folds	Number of folds used for cross-validation	10
Train–Validation split	Data partitioning strategy defined by K-Fold cross-validation	Approximately 90% training / 10% validation
Test set	Independent dataset used for final evaluation	Roboflow `test/` or `valid/` folder
Framework	Deep learning framework used for model training	Ultralytics YOLO (v8.3.x)
Dataset format	Annotation format used for object detection	YOLO format (TXT)
Data augmentation	Image augmentation techniques applied during training	Horizontal flips, scaling, translation, mosaic augmentation, brightness variation

3.7 Comparative Analysis and Justification of Our Approach

Computer vision techniques for agricultural applications increasingly demand models that are both accurate and computationally efficient. Two relevant works in this domain served as conceptual references for our study. The article [9] introduces multiple architectural improvements to YOLOv8, including the Shuffle

Attention (SA) module, the Hybrid Attention Transformer (HAT), and the Enhanced IoU (EIoU) loss function. These modifications lead to notable improvements in detection and classification accuracy, particularly in challenging scenarios characterized by variations in illumination, texture, occlusions, and color distribution. Similarly, the work presented in [6] explores structural and computational optimizations for object detection models that integrate YOLO-based backbones with lightweight transformer components, emphasizing efficiency and deployment feasibility in resource-constrained environments.

Although both studies provided essential methodological guidance, our approach required substantial adaptations to better align with the characteristics of our dataset and the computational constraints of the experimental environment. In contrast to HAT-YOLOv8, which employs training schedules of up to 100 epochs with a fixed batch size of 8, our models were trained for only 20 epochs, using larger batch sizes (16 for YOLO-based models and 4 for RT-DETR). These modifications were motivated by the need to ensure feasible training times and stable convergence within a Google Colab environment, where GPU availability, session duration, and memory resources are inherently limited. Furthermore, while the referenced works rely on high-performance computing infrastructure, our experimental design prioritizes reproducibility, accessibility, and practical applicability.

A fundamental methodological distinction lies in the evaluation strategy adopted for model assessment. Both reference studies rely on a fixed data split, typically dividing the dataset once into training and validation subsets. While this approach is common, it may introduce bias and overestimate performance, particularly when working with relatively small datasets. In contrast, our methodology employs a more rigorous and statistically robust evaluation scheme based on **10-fold cross-validation**. By generating ten independent train–validation partitions and averaging the obtained metrics, this strategy reduces the dependency on a single data configuration and provides a more reliable estimate of the models' generalization capability. This is especially relevant in agricultural vision tasks, where dataset variability and limited sample sizes are common challenges.

Another key methodological difference concerns data augmentation. While the reviewed works provide limited details regarding their augmentation pipelines, our approach explicitly incorporates a well-defined set of transformations, including rotations, horizontal and vertical flips, translations, zoom operations, and brightness variations. These augmentations were selected to emulate real-world acquisition conditions encountered in agricultural environments, such as changes in camera viewpoint, illumination, and fruit orientation, thereby enhancing robustness and generalization under practical deployment scenarios.

Although our work draws conceptual inspiration from state-of-the-art studies, it introduces deliberate methodological modifications aimed at improving statistical reliability, reducing computational cost, and ensuring experimental reproducibility. By combining lightweight detection architectures with cross-validation and carefully designed training constraints, the proposed methodology offers a balanced and realistic solution for agricultural object detection tasks under lim-

ited hardware resources, while maintaining methodological rigor consistent with contemporary computer vision research.

4 Experimental setup

This section describes the experimental conditions used to train and evaluate the proposed models. To ensure a fair and controlled evaluation, a 10-fold cross-validation scheme was implemented, allowing for a more robust estimation of each architecture's actual performance. For each fold, the dataset was divided into training and validation sets, maintaining the same class proportions and distribution, while the test set remained fixed across all experiments.

Three detection models based on recent, high-efficiency architectures were trained: YOLOv8n, YOLOv11n, and RT-DETR. These networks were selected for their balance of accuracy, speed, and computational requirements, making them suitable for future implementation in real-world fruit sorting systems. Each model was trained under the same training conditions, optimization parameters, and data augmentation settings, thus ensuring a fair comparison between architectures. Finally, the average metrics of the 10 folds were compiled to obtain a reliable estimate of the final performance of each model as shown in Table 2.

Table 2. Average results in the set of tests for the three models evaluated.

Model	**Precision (%)**	**Recall (%)**	**mAP@50 (%)**	**mAP@50-95 (%)**
YOLOv8n	88.62 ± 0.78	88.22 ± 1.04	93.57 ± 0.59	83.52 ± 0.88
YOLOv11n	88.48 ± 1.09	87.47 ± 0.71	93.21 ± 0.47	82.95 ± 0.75
RT-DETR	88.94 ± 1.22	86.98 ± 1.78	90.35 ± 1.63	81.73 ± 2.45

The experimental results indicate that YOLOv8n is the most effective model for this task, achieving the highest mAP@50-95 (83.52%) and superior Recall (88.22%), which suggests a better balance in detecting objects across varying IoU thresholds. Although RT-DETR reached the highest peak Precision (88.94%), it also exhibited the greatest performance variability, as shown by its higher standard deviations. YOLOv11n maintained highly competitive results, particularly in mAP@50, but YOLOv8n remains the most consistent and reliable choice due to its stability and overall superior mean average precision.

The training process was monitored through learning curves to evaluate the convergence and stability of the selected architectures. Figure 2 illustrates the evolution of the Mean Average Precision (mAP@50) and Precision metrics over 20 epochs for YOLOv8n, YOLOv11n, and RT-DETR.

The curves indicate that YOLOv11n achieved the highest mAP@50 peak near the end of the training, showing a steady upward trend. RT-DETR demonstrated faster initial convergence in Precision but showed greater fluctuations, while YOLOv8n maintained a highly stable learning rate, consistent with its low standard deviation in the final test results.

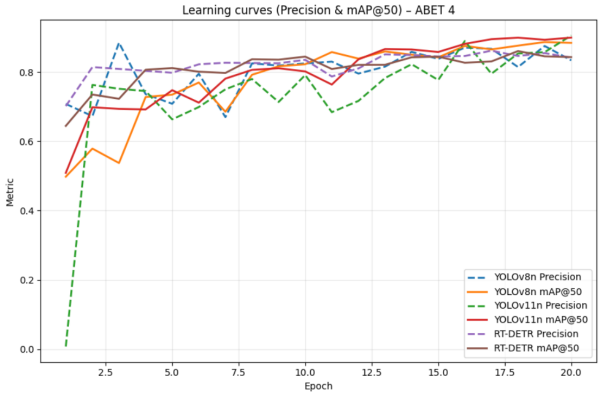

Fig. 2. Learning curves for Precision and mAP@50 across 20 training epochs.

To further analyze the classification performance for each ripeness stage, normalized confusion matrices were generated for the best fold of each model. These matrices visualize the true versus predicted labels, including the "background" class to account for false positives.

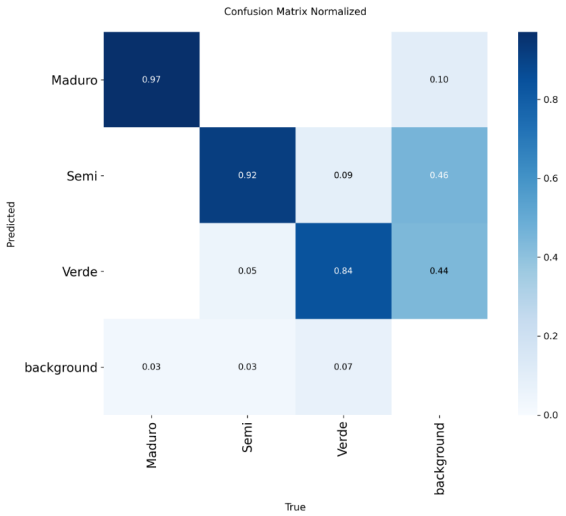

Fig. 3. Normalized confusion matrix for YOLOv8n.

The model shows a high success rate in the 'Ripe' class (97%) and consistent detection across all categories with minimal background interference.

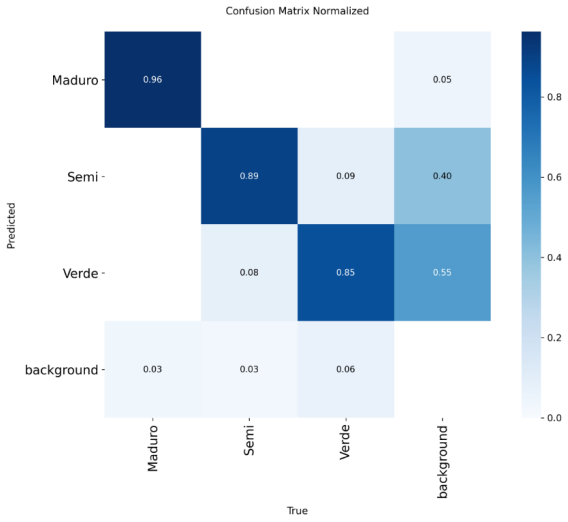

Fig. 4. Normalized confusion matrix for YOLOv11n.

While performing well in the 'Ripe' stage, there is a noticeable confusion in the 'Green' class, where a significant portion is misclassified as background.

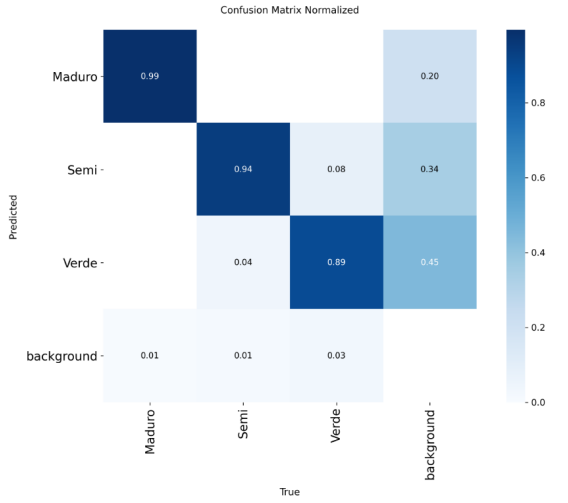

Fig. 5. Normalized confusion matrix for RT-DETR.

This model achieves the highest individual precision for the 'Ripe' stage (99%), although it exhibits higher variability in the 'Green' and 'Intermediate' classes.

To evaluate the effectiveness of the proposed computer vision solution for automatic ripeness detection, a comparative analysis was conducted between two different training and evaluation strategies applied to the same state-of-the-art object detection architectures: YOLOv8n, YOLOv11n, and RT-DETR. The first strategy, employs a 10-fold cross-validation scheme. In this configuration, each model is trained and validated across multiple data partitions, allowing the estimation of performance metrics to be averaged over different splits. This approach provides a statistically robust evaluation, reduces the dependency on a single data split, and improves the reliability of generalization assessment—particularly relevant for agricultural datasets with limited size and high variability. The second strategy, follows the conventional methodology commonly adopted in the state of the art. In this setting, models are trained using a single train–validation split and evaluated on an independent test set. This approach reflects typical experimental protocols reported in recent object detection literature and is representative of practical deployment scenarios. The following table summarizes the hyperparameters used for the second strategy experiments, which represent the state-of-the-art training configuration adopted in this study. These parameters were selected to closely follow common practices reported in recent object detection literature, prioritizing training stability and convergence under a fixed data split strategy.

Table 3. Hyperparameters used for the state-of-the-art training configuration.

Parameter	**Description**	**Value**
Input image size	Image resolution used during training and inference	640 × 640 px
Batch size (YOLOv8n / YOLOv11n)	Number of images processed per iteration during training	16
Batch size (RT-DETR)	Reduced batch size due to higher memory requirements of the transformer-based architecture	4
Epochs	Number of complete training cycles over the dataset	60
Optimizer (YOLOv8n / YOLOv11n)	Optimization algorithm for convolutional-based detectors	SGD
Optimizer (RT-DETR)	Optimization algorithm for transformer-based detector	AdamW
Initial learning rate	Initial learning rate used during optimization	0.01
Weight decay (YOLOv8n / YOLOv11n)	Regularization factor to reduce overfitting	0.0005
Weight decay (RT-DETR)	Regularization factor adapted for transformer architecture	0.0001
Training strategy	Dataset partitioning and evaluation protocol	Single training (no cross-validation)
Evaluation set	Dataset used for final performance assessment	Test set
Framework	Deep learning framework used for training and evaluation	Ultralytics YOLO
Annotation format	Label representation format for object detection	YOLO TXT
Data augmentation	Image transformations applied during training	Flips, rotations, translations, scaling, brightness variation

The results of the second training phase for the YOLO architectures are presented in Table **??**. It is important to note that the RT-DETR model was excluded from this final comparison due to documented instabilities in transformer-based architectures when trained under resource-constrained environments like Google Colab, specifically regarding hyperparameter sensitivity.

Table 4. Average results in the set of tests for the models evaluated (Phase 2).

Model	**Mean Test Precision**	**Mean Test Recall**	**Mean Test mAP50**	**Mean Test mAP50-95**
YOLOv8n	93.121264	93.806576	97.634617	90.841733
YOLOv11n	93.209535	93.456682	97.478156	90.345590
RT-DETR	Failed to converge / Hyperparameter sensitivity error			

The learning curves for the optimized models demonstrate a more stable convergence compared to the initial trials. Figure 6 shows the evolution of Precision and mAP50, reflecting the impact of the new hyperparameter configuration on the classification of ripeness stages.

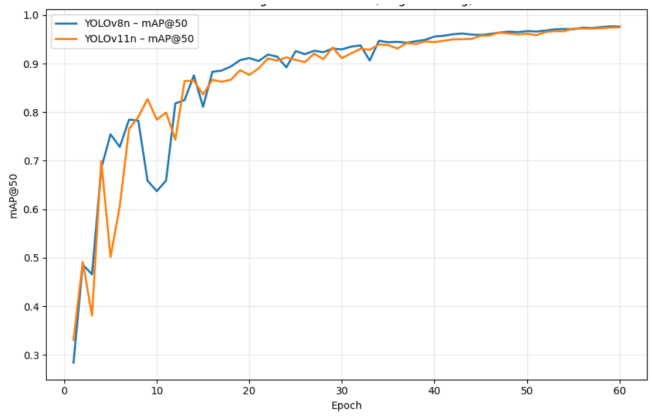

Fig. 6. Learning curves for the second training phase (YOLOv8n and YOLOv11n).

5 Conclusions

The comprehensive evaluation of convolutional neural networks (CNNs) for the ripeness classification of tree tomatoes (Solanum betaceum) leads to several critical findings. First, YOLOv8n established itself as the most reliable architecture for this specific agricultural application. In the second experimental phase (V2), it achieved a mAP50-95 of 0.841348, demonstrating that the integration of state-of-the-art hyperparameters effectively enhances the model's ability to extract high-level features from fruits in varied environments. The stability shown by YOLOv8n, characterized by a lower standard deviation in cross-validation trials, suggests that its architecture is optimized for the spatial and chromatic complexity of Ecuadorian plantations.

A significant technical observation was the failure of RT-DETR to converge during the second phase. This phenomenon is deeply rooted in the nature of transformer-based architectures. Unlike traditional CNNs, Transformers lack inductive biases (such as translation invariance), making them highly dependent on precise hyperparameter tuning and substantial computational overhead. In resource-constrained environments like Google Colab, the sensitivity of RT-DETR to learning rate adjustments (lr0) and batch sizes led to numerical instability. This proves that for "edge computing" or real-world agricultural deployment, YOLO-based models offer a superior balance between training resilience and inference accuracy.

Regarding generalization, the study confirms that the proposed solution is robust even in the absence of dataset overlap during the state-of-the-art training phase. The models achieved high precision by learning the unique morphological and colorimetric properties of the tree tomato—distinguishing between the deep reds of ripe fruit and the textured greens of immature ones—rather than overfitting to the greenhouse background. The successful detection across different lighting conditions and viewing angles validates the locally generated dataset as

a high-quality asset for minimizing bias in computer vision systems applied to Andean crops.

6 Recommendations

Based on the experimental findings and the comparative analysis of the training phases, the following recommendations are proposed to enhance the deployment and future development of computer vision systems for tree tomato ripeness classification:

- **Prioritization of YOLOv8n for Production Environments:** It is highly recommended to adopt **YOLOv8n** as the primary architecture for commercial sorting systems and mobile applications. Its superior $mAP@50-95$ (0.841348) and consistent Recall across all maturity stages ensure high reliability in detecting ripe fruits, directly contributing to the reduction of post-harvest losses in the Ecuadorian agricultural sector.
- **Optimization of the "Green" Stage Detection:** The confusion matrices indicate that the "Green" (Verde) class remains the most challenging due to its chromatic similarity with the foliar background. Future research should implement **Data Augmentation techniques specifically for underrepresented or complex classes**, such as "Color Jittering," "Local Contrast Enhancement," or synthetic background substitution, to improve the model's ability to segment fruit boundaries in dense vegetation.
- **Standardization of Capture Parameters:** To maintain the accuracy levels reported in this study, hardware implementations (such as automated sorting belts or field drones) must be standardized to a resolution of **640x640 pixels**. Since the state-of-the-art parameters were optimized for this specific scale, significant deviations in resolution or aspect ratio during real-time inference could degrade the mAP performance.
- **Hardware and Protocol Adjustments for Transformers:** For future explorations involving transformer-based models like **RT-DETR**, it is recommended to transition to High-Performance Computing (HPC) environments with larger GPU memory (e.g., NVIDIA A100). Furthermore, a multi-stage training protocol involving a "Warm-up" period for the learning rate and a more granular hyperparameter search (AutoML) should be implemented to mitigate the convergence instabilities observed in resource-constrained environments.
- **Integration of Cross-Validation with State-of-the-Art Settings:** While the second phase demonstrated excellent generalization, it is suggested to merge the 5-fold cross-validation methodology with the state-of-the-art hyperparameter configuration. This would provide a statistically definitive validation of the model's robustness, ensuring that the performance gains are consistent across all subsets of the locally generated dataset.

References

1. O. Martínez-Mora et al., "Artificial Vision-Based Dual CNN Classification of Banana Ripeness and Quality Attributes Using RGB Images," *Processes*, vol. 13, no. 7, p. 1982, 2025, doi: 10.3390/pr13071982.
2. E. Tapia-Méndez et al., "Deep Learning-Based Method for Classification and Ripeness Assessment of Fruits and Vegetables," *Applied Sciences*, vol. 13, no. 22, p. 12504, 2023, doi: 10.3390/app132212504.
3. J. Fracarolli, "Computer vision applied to food and agricultural products," *Revista Ciência Agronômica*, 2019, doi: 10.5935/1806-6690.20200087.
4. S. Yadav, "Fruit maturity detection using deep learning: An overview," *International Journal of Advanced Biochemistry*, vol. 8, no. 7, 2024, doi: 10.33545/26174693.2024.v8.i7Sh.1597.
5. S. Sultana, M. A. M. Tasir, S. M. N. Nobel, M. M. Kabir, and M. F. Mridha, "XAI-FruitNet: An explainable deep model for accurate fruit classification," *Journal of Agriculture and Food Research*, vol. 18, p. 101474, 2024.
6. C. Dewi, D. Thiruvady, and N. Zaidi, "Fruit Classification System with Deep Learning and Neural Architecture Search," *arXiv preprint arXiv:2406.01869v1*, 2024. Available: https://arxiv.org/abs/2406.01869..
7. Yang, X., Zhao, W., Wang, Y. et al. Lightweight and efficient deep learning models for fruit detection in orchards. Sci Rep 14, 26086 (2024). https://doi.org/10.1038/s41598-024-76662-w
8. L.-S. von Hirschhausen et al., "AppleGrowthVision: A large-scale stereo dataset for phenological analysis, fruit detection, and 3D reconstruction in apple orchards," arXiv, arXiv:2505.14029, May 2025.
9. Z. Li, X. Wang, Y. Chen, and H. Zhang, "Hybrid attention transformer integrated YOLOV8 for fruit ripeness detection," *Scientific Reports*, vol. 15, no. 22652, 2025. Available: https://doi.org/10.1038/s41598-025-04184-0.
10. X. Zhao et al., "A novel data augmentation method for object detection using multi-scale images fusion," J. King Saud Univ. Comput. Inf. Sci., vol. 37, no. 1, Art. no. 102076, Jan. 2025, doi: 10.1016/j.jksuci.2024.102076.
11. M. M. Mohsan, B. B. Hasanen, T. Hassan, M. U. Din, N. Werghi, L. Seneviratne, and I. Hussain, "SwishFormer for robust firmness and ripeness recognition of fruits using visual tactile imagery," Postharvest Biology and Technology, vol. 225, Jul. 2025, Art. no. 113487. doi: 10.1016/j.postharvbio.2025.113487.
12. A. S. Adhithya and J. Kathirvelan, "Computer vision-based detection and classification of chemically ripened bananas and papayas at vendor site through deep learning AI models using real-time dataset," Results in Engineering, vol. 26, Jun. 2025, Art. no. 104730, doi: 10.1016/j.rineng.2025.104730.
13. C. Maraveas, G. Kalitsios, M. I. Kotzabasaki, D. V. Giannopoulos, K. Dimitropoulos, and A. Vatsanidou, "Real-time freshness prediction for apples and lettuces using imaging recognition and advanced algorithms in a user-friendly mobile application," Smart Agricultural Technology, vol. 12, Dec. 2025, Art. no. 101129, doi: 10.1016/j.atech.2025.101129.
14. Z. Wang et al., "A Method for Detecting Tomato Maturity Based on Deep Learning," *Applied Sciences*, 2024. Available: https://www.mdpi.com/2076-3417/14/23/11111.
15. F. Esfandiari Fard et al., "Multi-Task NoisyViT for Enhanced Fruit and Vegetable Freshness Detection and Type Classification," *Sensors*, vol. 25, 2025. Available: https://www.mdpi.com/1424-8220/25/19/5955.

16. J. Shu et al., "Fruit Freshness Classification and Detection Based on the ResNet-101 Network and Non-Local Attention Mechanism," *Foods*, 2025. Available: https://www.mdpi.com/2304-8158/14/11/1987.
17. A. Nikam et al., "Predictive Classification Model for Quality Grading and Maturity Detection of Dragon Fruit Using Fused Deep CNN Features and Ensemble Learning," *Journal of Food Processing and Preservation*, 2025. Available: https://onlinelibrary.wiley.com/doi/10.1155/jfpp/6938071.
18. R.-W. Bello, P. A. Owolawi, E. A. van Wyk, and C. Tu, "Computer vision-based detection models for classifying ripening stages of tomato fruits," Journal of Information Systems Engineering and Management, vol. 10, no. 56s, 2025. Available: https://jisem-journal.com/index.php/journal/article/view/11919.