High-Precision Orange Leaf Disease Detection Using Edge-Optimized Lightweight YOLO nano models

Mohamed Amine Bouallegui[1], Imen Saidi [2], Amine Abadi[3], and David Fofi[4]

***Abstract*— Orange leaf diseases pose a significant threat to agricultural productivity and crop quality, making early and accurate detection essential for reducing economic losses. Recent advances in deep learning have enabled automated plant disease detection with high accuracy; however, achieving reliable performance across visually similar and imbalanced disease classes remains a challenge. In this paper, a deep learning–based orange leaf disease detection framework is presented using different lightweight YOLO object detection models, with a particular focus on YOLO26n. A custom orange leaf disease dataset is constructed and extensively preprocessed, including auto-orientation, resizing, and data augmentation, to enhance robustness and generalization. Multiple YOLO architectures—YOLO8n, YOLO11n, YOLO12n, and YOLO26n—are trained under identical experimental conditions to ensure a fair and consistent comparison. Model performance is evaluated using standard detection metrics, including precision, recall, F1-score, mean average precision, and class-wise confusion matrix analysis. Experimental results demonstrate that all evaluated YOLO models achieve high and stable detection performance across disease categories, confirming the effectiveness of lightweight YOLO architectures for orange leaf disease analysis. Notably, YOLO12n and YOLO26n show more balanced class-wise predictions and smoother training convergence, indicating improved learning stability and robustness compared to other models. The comparative analysis provides valuable insights into the impact of different YOLO model architectures on detection accuracy and robustness, supporting informed model selection for practical agricultural disease detection applications.**

I. INTRODUCTION

Orange crops are among the most widely cultivated fruit crops worldwide, playing a key role in local economies, employment, and the global food supply chain. Their high nutritional and economic value makes them a staple of both smallholder and commercial agriculture. However, orange plants are highly susceptible to a variety of leaf diseases, which can severely reduce photosynthesis, hinder fruit development, and ultimately lower overall yield. Common diseases such as orange canker, black spot, and huanglongbing (greening) often manifest early on the leaves, producing visible symptoms like necrotic spots, chlorosis, and lesions. These early leaf symptoms make leaf-based analysis an effective and practical approach for timely disease diagnosis.

Traditional detection methods in orchards rely heavily on manual inspection by experienced agricultural experts. While such methods can be accurate under controlled conditions, they are labor-intensive, time-consuming, and prone to human error. Moreover, in large-scale plantations or remote farms, frequent inspections are often impractical, leading to delayed detection and increased risk of disease spread. These limitations highlight the urgent need for automated, reliable, and scalable disease detection systems capable of supporting timely agricultural decision-making.

In recent years, the convergence of deep learning and computer vision has revolutionized plant disease diagnosis. Convolutional neural networks (CNNs) have demonstrated remarkable ability to automatically extract complex visual features from raw leaf images, enabling robust classification even under varying lighting conditions and complex backgrounds. Beyond simple classification, object detection frameworks provide the added advantage of localizing disease regions within the leaf, which is critical for precise monitoring and quantification of disease severity.

Among modern object detection models, the You Only Look Once (YOLO) family stands out due to its single-stage detection paradigm [1], [2], which combines high detection accuracy with fast inference speeds. Unlike multi-stage detectors, YOLO models predict bounding boxes and class probabilities in a single forward pass, making them highly efficient for real-time applications in agriculture, such as on-field monitoring using drones or edge devices. Successive versions of YOLO, including YOLO8, YOLO11, YOLO12, and YOLO26, have progressively introduced enhancements in backbone architectures, feature pyramid networks, training strategies, and loss functions. These improvements aim to improve detection accuracy, robustness to occlusion and overlapping symptoms, and overall computational efficiency. Additionally, lightweight variants have been developed to reduce model size and inference time, making them suitable for deployment on resource-constrained devices such as mobile phones, embedded systems, or unmanned aerial vehicles [9], [10].

Despite these advances, the practical effectiveness of YOLO-based disease detection systems depends on several factors, including the quality and diversity of the training dataset, appropriate data augmentation, optimal hyperparameter tuning, and careful model optimization. Furthermore, comparative studies evaluating different YOLO versions un-

[1]Mohamed Amine Bouallegui is with University of Tunis El Manar, National Engineering School of Tunis, Laboratoire de Recherche en Automatique, Tunis, Tunisia mohamedamine.bouallegui@etudiant-enit.utm.tn

[2]Imen Saidi is with University of Tunis El Manar, National Engineering School of Tunis, Laboratoire de Recherche en Automatique, Tunis, Tunisia imen.saidi@gmail.com

[3]Amine Abadi is with Université Bourgogne Europe, Laboratory ImViA EA 7535, University of Bourgogne, 21000 Dijon, France Dijon, France Amine.Abadi@u-bourgogne.fr

[4]David Fofi is with Université Bourgogne Europe, Laboratory ImViA EA 7535, University of Bourgogne, 21000 Dijon, France david.fofi@u-bourgogne.fr

der consistent agricultural datasets remain limited, particularly for real-world orange orchards with varying environmental conditions. Addressing these gaps is essential for developing robust, scalable, and deployable automated solutions that can enhance crop health monitoring and support sustainable agricultural practices [3], [4].

II. Literature Review

Deep learning has become the dominant paradigm for automated plant disease detection due to its ability to automatically learn complex and discriminative visual features from raw images. Early works primarily relied on convolutional neural network (CNN)–based classification models [5], [6], which map an input image $x \in \mathbb{R}^{h\times w\times 3}$ to a categorical label $y \in \{1,\dots,C\}$:

$$\hat{y} = f_\theta(x) \tag{1}$$

Where f_θ denotes the neural network parameterized by weights θ, and $\hat{y} \in [0,1]^C$ is the predicted probability distribution over C classes. The network is trained by minimizing the cross-entropy loss:

$$\mathcal{L}_{CE}(\theta) = -\sum_{i=1}^{C} y_i \log \hat{y}_i \tag{2}$$

While CNN classifiers achieve high accuracy in controlled settings, they cannot localize disease regions and often fail under complex backgrounds, varying illumination, and overlapping symptoms. To address this, object detection frameworks are employed, predicting both the location and class of disease symptoms.

Modern single-stage detectors, such as the YOLO family, use a composite loss function combining localization, objectness, and classification components:

$$\mathcal{L}_{total} = \lambda_{loc}\mathcal{L}_{loc} + \lambda_{obj}\mathcal{L}_{obj} + \lambda_{cls}\mathcal{L}_{cls} \tag{3}$$

The backbone extracts hierarchical feature maps F_l at different levels l:

$$F_l = \phi_l(F_{l-1}) \tag{4}$$

Where ϕ_l represents convolution, batch normalization, and activation layers. The neck fuses multi-scale features to improve detection of small and large disease patterns [3], [4].

A. Comparison of YOLO Variants

TABLE I
COMPACT TECHNICAL COMPARISON OF YOLO VARIANTS

YOLO Ver	Backbone	Neck	Params
YOLO8	CSPDarknet53	PANet	31M
YOLO11	CSPDarknet53	PANet	34M
YOLO12	CSPDarknet53	PANet + Fusion	36M
YOLO26	Lightweight CSP	Enhanced PANet	12M

B. Evaluation Metrics

Precision-recall at box level (PR_{box}) is defined as:

$$\text{Precision}_{box} = \frac{TP_{box}}{TP_{box} + FP_{box}} \tag{5}$$

$$\text{Recall}_{box} = \frac{TP_{box}}{TP_{box} + FN_{box}} \tag{6}$$

Mean average precision at IoU thresholds is computed as:

$$\text{mAP@0.5} = \frac{1}{C}\sum_{i=1}^{C} AP_i^{0.5} \tag{7}$$

$$\text{mAP@0.5:0.95} = \frac{1}{C}\sum_{i=1}^{C}\frac{1}{10}\sum_{t=0.5}^{0.95} AP_i^{t} \tag{8}$$

Lightweight YOLO26 is particularly suited for edge AI applications due to reduced parameters and faster inference [7], [8], while earlier variants prioritize accuracy on high-end GPU systems. However, comparative studies under consistent datasets remain limited. This work systematically evaluates these YOLO variants with emphasis on YOLO26, analyzing accuracy, stability, and efficiency for practical agricultural deployment.

III. Methodology

This section details the methodology employed to develop and evaluate the edge AI-based orange leaf disease detection system. The overall pipeline consists of four main phases: (1) dataset preparation and annotation, (2) model training and optimization, (3) model conversion for edge deployment, and (4) performance evaluation on the target hardware. A systematic approach was adopted to ensure fair comparison across different YOLO architectures under identical training and deployment conditions.

A. Dataset Description and Preparation

The dataset used in this study consists of 6,615 images of orange leaves exhibiting various disease conditions. The images were collected from multiple sources to ensure diversity in lighting conditions, backgrounds, and leaf orientations. The dataset encompasses 10 distinct disease classes and one healthy class, as detailed in Table II.

TABLE II
DATASET CLASS DISTRIBUTION

Disease Class	Number of Images
Orange Canker	588
Orange Die Back	642
Orange Foliage Damaged	632
Orange Greening	254
Orange Healthy Leaves	1,349
Orange Mealybugs	603
Orange Powdery Mildew	598
Orange Shot Hole	560
Orange Spiny Whitefly	672
Orange Yellow Dragon	407
Orange Yellow Leaf	310
Total	**6,615**

A custom automated annotation pipeline was developed to generate bounding boxes for all leaf images. The annotation system utilized computer vision techniques implemented with OpenCV to detect leaf boundaries based on pixel color differentials between the leaf and background regions. This approach ensured consistent and accurate labeling across the dataset, providing high-quality ground truth for model training.

Fig. 1 illustrates the annotation style and bounding box placement used throughout the dataset.

Fig. 1. Example of Orange leaf annotation showing bounding boxes around disease regions.

B. Data Preprocessing and Augmentation

The dataset was split into training and validation subsets using a 92:8 ratio, resulting in 6,094 training images and 521 validation images, with no separate test set; final evaluation was performed on the entire dataset after training to maximize data utilization. Prior to training, images underwent preprocessing to standardize inputs and improve model robustness, including auto-orientation for consistent spatial alignment, and resizing to 512×512 pixels to match network input dimensions.

Data augmentation was applied to enhance diversity and generalization, addressing variations in lighting, leaf orientation, background, and acquisition devices. Each training sample was augmented with three operations: random rotation within $[-15°, +15°]$, saturation adjustment between -31% and $+31\%$, and brightness adjustment between -25% and $+25\%$. These steps expanded the dataset to 17,211 images, as summarized in Table III.

After augmentation, the dataset size increased to a total of 17,211 images, distributed as shown in table III.

TABLE III
DATASET SPLIT AFTER AUGMENTATION

Set	Percentage	Number of images
training	92%	15,894
validation	8%	1,317
test	0%	0
total	100%	17,211

No separate test set was allocated, as performance evaluation was carried out on the complete dataset following training. this strategy ensures maximum data availability for model learning while preserving a representative validation subset for monitoring convergence and performance during training. The class-wise distribution of the augmented dataset is illustrated in Fig. 2. the figure reveals a moderate level of class imbalance across the different orange leaf disease categories, which further motivates the use of data augmentation to improve class representation and learning stability.

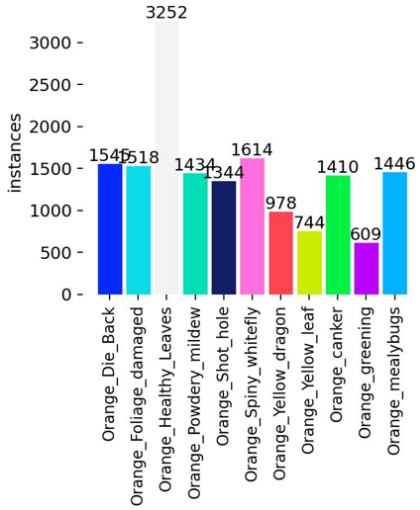

Fig. 2. class-wise distribution of orange leaf images in the dataset

C. Model Architecture and Training Configuration

Four state-of-the-art YOLO architectures were evaluated: YOLO8, YOLO11, YOLO12, and YOLO26. All models were initialized with official pre-trained weights and trained

under identical settings. The main training hyperparameters are summarized in Table IV.

The complete system was implemented and trained using Python within the Google Colab environment. Model performance was evaluated using both detection-based metrics and overall accuracy measures to provide a comprehensive assessment of classification and localization capabilities.

TABLE IV
TRAINING HYPERPARAMETERS FOR YOLO MODELS

Hyperparameter	Value
Input Image Size	640 × 640 px
Batch Size	16
Workers	4
Device	0
Optimizer	AdamW
Initial Learning Rate (lr_0)	0.0001
Final LR Factor (lrf)	0.1
Momentum	0.937
Weight Decay	0.0005
Warm-up Epochs	3.0
Warm-up Momentum	0.8
Warm-up Bias LR	0.1
Epochs	100
Box Loss Weight	7.5
Class Loss Weight	0.5
DFL Loss Weight	1.5
HSV Augmentation (H, S, V)	0.015, 0.7, 0.4
Rotation (degrees)	0.0
Translation	0.1
Scale	0.5
Shear	0.0
Perspective	0.0
Horizontal Flip	0.5
Vertical Flip	0.0
Mosaic	1.0
MixUp	0.0
Copy-Paste	0.0
Half Precision	False
Deterministic	False

D. Experimental Design

A controlled experimental setup was established to ensure reproducible and comparable results across all evaluated models. Training consistency was maintained by using identical dataset splits, data augmentation strategies, and hardware resources for all experiments. Inference was standardized by evaluating all models under the same input conditions, including consistent lighting, background, and object distance. To ensure statistical reliability, each performance measurement was repeated ten times, with mean values and corresponding standard deviations reported. Finally, a comparative analysis was conducted to examine trade-offs between detection accuracy and inference speed, enabling the identification of optimal configurations for real-time edge deployment.

This systematic methodology enables a comprehensive evaluation of YOLO architectures for real-time orange leaf disease detection on edge devices, while providing practical insights into deployment considerations for precision agriculture applications.

IV. RESULTS AND DISCUSSION

This section presents the experimental results obtained from training and evaluating four YOLO architectures (YOLO8n, YOLO11n, YOLO12n, and YOLO26n) on the orange leaf disease dataset. The analysis focuses on detection accuracy, model efficiency, and edge deployment suitability. In addition to standard performance metrics, confusion matrices are used to provide a class-wise evaluation of predicted versus ground-truth labels, enabling a detailed analysis of true positives, false positives, true negatives, and false negatives for each disease class.

A. Class-wise Detection Analysis

The confusion matrix is a common evaluation tool for classification and object detection, showing the relationship between true labels and model predictions. Rows represent actual classes, columns represent predicted classes, diagonal elements indicate correct predictions, and off-diagonal entries show misclassifications. It provides detailed class-wise performance insights and highlights inter-class confusion not captured by overall metrics like accuracy or mAP.

Fig. 3 presents the class-wise detection performance of the evaluated YOLO8n models.

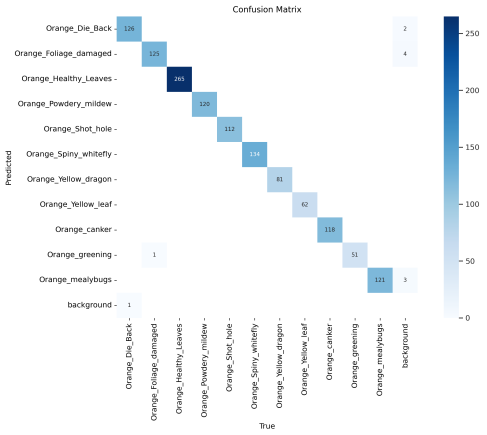

Fig. 3. Normalized confusion matrix of YOLO8n

Fig. 4 presents the class-wise detection performance of the evaluated YOLO11n models.

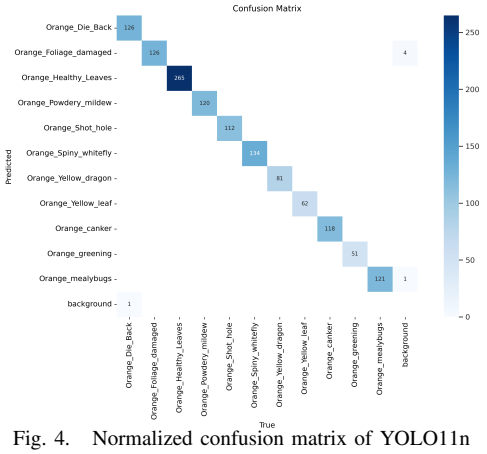

Fig. 4. Normalized confusion matrix of YOLO11n

Fig. 5 presents the class-wise detection performance of the evaluated YOLO12n models.

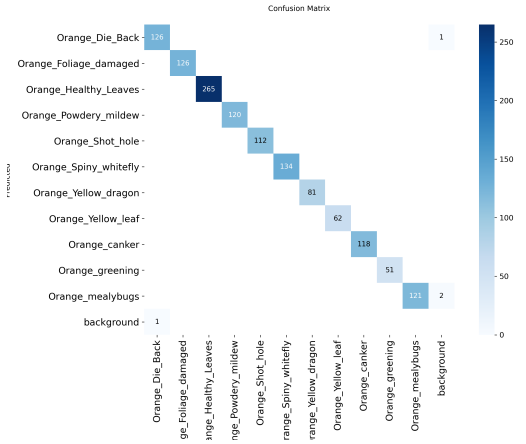

Fig. 5. Normalized confusion matrix of YOLO12n

Fig. 6 presents the class-wise detection performance of the evaluated YOLO26n models.

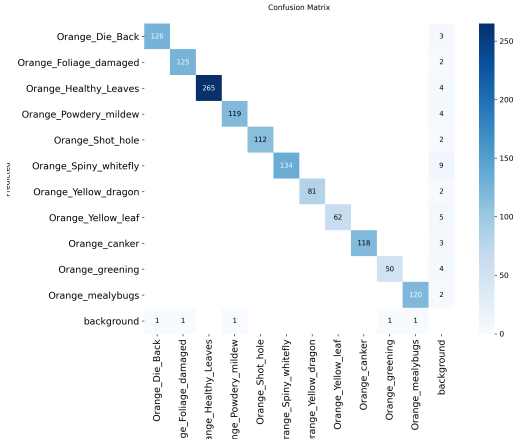

Fig. 6. Normalized confusion matrix of YOLO26n

B. Training and Convergence Analysis

Training curves provide a detailed and quantitative representation of a model's learning process by illustrating the evolution of key performance metrics over the course of training. By visualizing these curves, researchers can gain valuable insights into the behavior of the model at different stages, including how quickly it converges, how well it generalizes, and whether any irregularities occur during training. These curves track both training and validation losses, allowing the observation of how effectively the model is optimizing its objective function and how well it performs on unseen data. Additionally, they can reveal potential issues such as overfitting, underfitting, or unstable learning dynamics early in the training process.

In object detection models such as YOLO, the monitored losses typically include the box loss, which reflects the accuracy of predicted bounding box coordinates (decreasing from $\sim$1.0–1.2 to $\sim$0.4), the classification loss (cls loss, decreasing from $\sim$1.5 to $\sim$0.3), which measures the correctness of predicted class labels, and the distribution focal loss (dfl loss, decreasing from $\sim$1.6 to $\sim$1.0), which evaluates the precision of bounding box distributions. Tracking these different components of loss provides a more granular understanding of the model's performance and helps identify whether issues arise from localization, classification, or bounding box regression.

In addition to loss values, training curves also track evaluation metrics that reflect detection performance and overall model effectiveness. These metrics include precision, which stabilizes near 0.97, indicating a low false positive rate; recall, reaching approximately 0.96, showing that most true instances were correctly detected; mAP50, increasing to $\sim$0.95, representing the mean average precision at 50% IoU; and mAP50–95, rising to $\sim$0.90, capturing the model's performance across stricter IoU thresholds. Monitoring these metrics allows for a comprehensive understanding of both localization and classification quality, as well as how the model balances false positives and false negatives across different classes and instances.

The shape, smoothness, and trends of these curves provide additional insight into the learning dynamics. Smooth, consistently decreasing loss curves and steadily improving evaluation metrics suggest that the model is effectively optimizing its parameters, is learning relevant features from the data, and is likely to generalize well to unseen images. On the other hand, oscillations, spikes, or irregular patterns in the curves can indicate potential problems such as unstable gradient updates, overfitting, or underfitting. Such patterns can inform the need for adjustments in learning rates, regularization techniques, batch sizes, or other hyperparameters to stabilize training and improve performance.

Careful examination of training curves in conjunction with evaluation metrics is therefore essential not only for assessing the final performance of a model but also for understanding the efficiency, stability, and overall quality of the learning process. Analyzing these curves over multiple training runs can reveal trends such as convergence speed, class-wise learning behavior, sensitivity to hyperparameter changes, and robustness to dataset variations. This information can guide informed decisions regarding model selection, hyperparameter tuning, early stopping criteria, and deployment readiness. Ultimately, such analysis ensures that the trained object detection system performs reliably and consistently in real-world agricultural scenarios, where environmental variability and challenging visual conditions can significantly affect detection accuracy and robustness.

Fig. 7 presents the training and convergence behavior of the evaluated YOLO8n architecture:

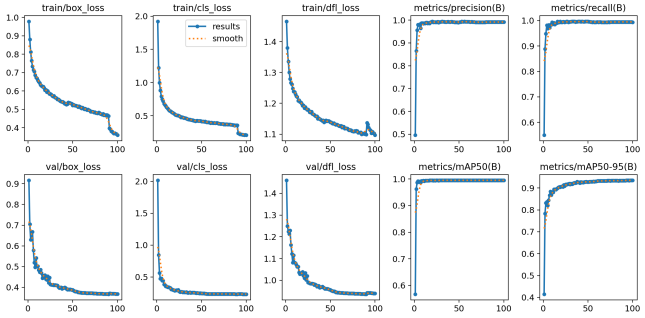

Fig. 7. Overall Training Result for YOLO8n

Fig. 8 presents the achieved results values for the YOLO11n architecture:

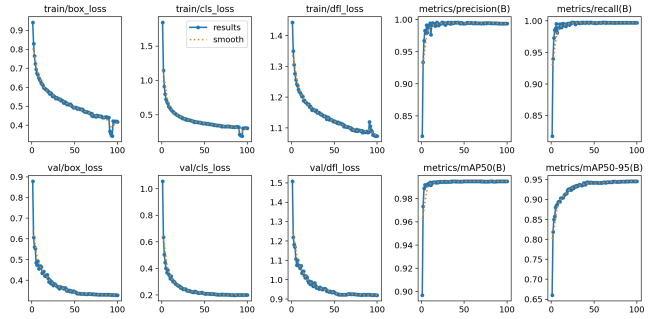

Fig. 8. Overall Training Result for YOLO11n

Likewise, Fig. 9 presents the results for the YOLO12n architecture:

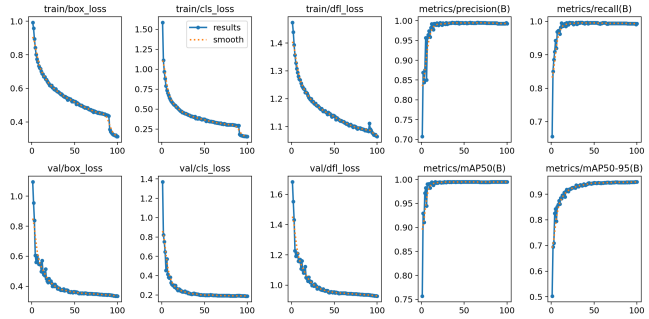

Fig. 9. Overall Training Result for YOLO12n

And finally, Fig. 10 presents the results for the YOLO26n architecture:

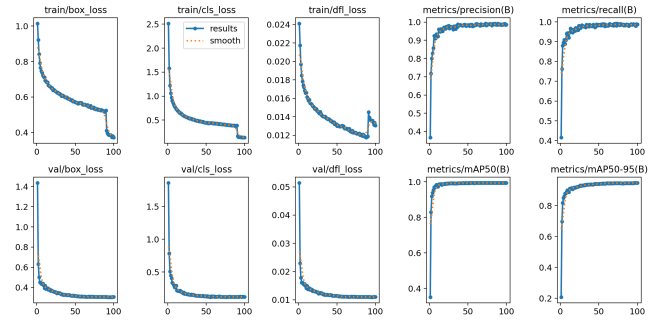

Fig. 10. Overall Training Result for YOLO26n

CONCLUSION

in this study, four lightweight architectures—YOLO8n, YOLO11n, YOLO12n, and YOLO26n—were systematically evaluated for orange leaf disease detection using a curated dataset of 6,615 annotated leaf images, augmented to 17,211 samples to improve class balance and model generalization. all models were trained under identical settings, including a 640 × 640 input resolution, batch size of 16, adamw optimizer, 100 epochs, and comprehensive data augmentation (hsv, geometric, and mosaic transformations). these controlled conditions ensured a fair comparison across architectures and provided a reproducible framework for future research.

all evaluated models achieved exceptional performance, exceeding 99% mAP@50, confirming that lightweight YOLO variants are highly effective for high-precision agricultural tasks. among them, YOLO12n emerged as the top performer for accuracy-critical applications, attaining the highest mAP@50:95 (94.986%) and the lowest gpu latency (1.64 ms). YOLO11n continues to serve as a robust benchmark for efficiency, while YOLO26n demonstrated the greatest promise for edge deployment: its nms-free architecture achieved the fastest cpu inference (38.9 ms), representing a 51% speed improvement over the YOLO8n baseline.

These findings emphasize the practical use of precision agriculture systems: YOLO12n is best suited for GPU-based environments requiring high accuracy, while YOLO26n offers a lightweight and fast solution for CPU-based edge devices and real-time field monitoring. The use of standardized training settings and augmentation methods also ensures reliable and reproducible results. future work will extend these evaluations to real-world orange orchards, focusing on mobile and embedded platforms to enable real-time, on-device disease detection and management. additionally, further exploration of hybrid training strategies, incremental learning for evolving disease patterns, and model compression techniques may enhance deployment feasibility while maintaining high detection precision, supporting sustainable and scalable precision agriculture practices.

ACKNOWLEDGMENT

This work was supported by the EIPHI Graduate School (contract ANR-17-EURE-0002) and the Bourgogne-Franche-Comté Region).

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779-788.

[2] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7263-7271.

[3] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[4] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv preprint arXiv:2004.10934*, 2020.

[5] G. Jocher, A. Chaurasia, A. Stoken, et al., "Ultralytics YOLOv8: State-of-the-Art Object Detection and Image Segmentation Model," *Ultralytics GitHub Repository*, 2023.

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

[7] C. Li, L. Li, H. Jiang, et al., "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," *arXiv preprint arXiv:2209.02976*, 2022.

[8] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 7464-7475.

[9] Y. Zhang, H. Wang, J. Chen, and L. Zhang, "YOLOv10: Real-Time End-to-End Object Detection with Dual-Label Assignment and Efficiency-Driven Design," *arXiv preprint arXiv:2405.14458*, 2024.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Amsterdam, The Netherlands, Oct. 2016, pp. 21-37.