Edge AI rice disease detection: A hardware performance comparison

Chu Minh Quang[1], Nguyen Hung Cuong[1], Mai Ngoc Tam[1], Vu Duc Long[1], Ngo Phuong Thuy[3], Nguyen Huy Phuong[1], Bui Dang Thanh[1,2] and Trinh Cong Dong[2,*]

[1]*School of Electrical and Electronic Engineering, Hanoi University of Science and Technology*
[2]*Institute of Automation and Control Technology, Hanoi University of Science and Technology*
[3]*Faculty of Engineering, Vietnam National University of Agriculture*
[*]*Corresponding author E-mail: dong.trinhcong@hust.edu.vn*
DOI: https://doi.org/10.64032/mca.v30i2.410

Abstract

This study addresses the challenge of real-time rice disease detection under resource-constrained conditions by developing an edge AI system for rice leaf disease identification without cloud dependency. A lightweight YOLOv5n object detection model was trained on a rice disease dataset and optimized using post-training quantization. The quantized model was deployed on an ultra-low-power microcontroller (MCU) integrated with an Arm Ethos-U55 Neural Processing Unit (NPU), and its performance was compared against a Raspberry Pi 4 and a workstation with a high-performance CPU. Results show that the quantized model maintains high detection accuracy (mAP > 90%) and achieves real-time inference on the microcontroller (around 16 FPS) at only 1.53 W, roughly 10 times faster and with 54% lower power consumption compared to the Raspberry Pi. While the CPU performance reached the fastest inference (9.5 ms), its energy consumption was significantly higher. In conclusion, the research demonstrates the feasibility of deploying quantized vision models on low-power edge devices for smart agriculture. The findings highlight the trade-offs between performance and energy efficiency, marking a successful implementation of quantized YOLOv5n on a microcontroller NPU in the smart agricultural sector.

Keywords: *Edge AI; Embedded systems; Quantization; Rice disease detection; YOLOv5n.*

Abbreviations

CNN	Convolutional Neural Network
FPS	Frame per second
NPU	Neural Processing Unit
CPU	Central Processing Unit
RPi	Raspberry Pi
FP32	32-bit floating-point
INT8	8-bit integer

1. Introduction

Rice serves as a staple food source for over half of the global population and stands as a primary driver of economic stability in Southeast Asia. This is especially true in Vietnam, which consistently ranks among the world's top three rice exporters alongside Thailand and India, proving that rice production is a determinant of the nation's economic sustainability.

Beyond ensuring domestic food security, rice production has evolved into a strategic economic sector that contributes to national stability and provides employment for millions of rural workers [1]. However, rice cultivation is strongly affected by biotic and abiotic stresses. The tropical climate of the region creates favorable conditions for fungal and bacterial diseases to thrive, posing severe risks to crop productivity. Rice blast epidemics can cause yield losses ranging from 10% to 30%, with severe outbreaks in untreated fields potentially leading to near-total crop failure [2]. Therefore, early detection of rice diseases is critical for minimizing yield loss and reducing the excessive use of chemical treatments. Traditionally, disease diagnosis in rice fields relies on visual inspection by farmers or agricultural specialists, which is often slow, subjective and difficult to apply over large farming areas. These limitations highlight the need for automated, in-field tools that can assist farmers with rapid and reliable disease recognition. Advances in computer vision and deep learning have demonstrated that images of crop leaves can be used to train highly accurate plant disease classifiers on large datasets. Yet, many proposed solutions depend on cloud-based processing, which introduces latency, demands stable internet connectivity and raises privacy concerns, thereby limiting their viability in rural agricultural environments.

In recent years, deep learning techniques, particularly convolutional neural networks, have been widely applied to automatic plant disease detection because of their strong ability to extract relevant features from images and achieve high classification accuracy. Early studies employed traditional machine learning methods such as Support Vector Machines (SVM), Naïve Bayes classifiers, and simple CNN architectures to detect rice leaf diseases. While these approaches demonstrated that image-based disease detection is feasible, their performance was often unstable in real field conditions. Variations in lighting, background complexity, and disease severity frequently reduced detection accuracy [3].

With the advancement of object detection techniques, one-stage detectors from the YOLO family have gained popularity

Received: 23 January 2026; Revised: 26 February 2026; Accepted: 4 March 2026.

in agricultural applications due to their fast inference speed. Gao *et al.* [4] proposed an improved YOLOv5 model for rice leaf disease detection and reported superior performance compared to traditional CNN classifiers and two-stage detectors such as Faster R-CNN. Similarly, Cheng *et al.* [5] introduced an improved YOLOv7-Tiny model that achieved higher detection accuracy while reducing the model size, making it more suitable for devices with limited hardware resources. These studies indicate that YOLO-based models are effective for real-time crop disease detection.

Although CNN-based models can achieve high accuracy in controlled experimental settings, their deployment in real agricultural environments remains challenging. Limited computational resources, low energy availability, and unstable network connectivity are common constraints in field conditions. To address these challenges, Edge Artificial Intelligence (Edge AI) has been proposed, in which data processing and inference are performed directly on embedded or edge devices rather than on remote cloud servers. Several studies have successfully deployed lightweight deep learning models on edge platforms such as Raspberry Pi and other embedded processors. To reduce memory usage and inference time, model optimization techniques including quantization, pruning, and precision adjustment are commonly applied while maintaining acceptable accuracy [6]. These results demonstrate that deep learning–based rice disease detection can be implemented on low-resource devices.

This paper proposes and assesses a deep learning-based rice disease detection system deployed on multiple hardware platforms. A single trained model is ported to several representative platforms and their performance is systematically compared in terms of inference latency, resource usage, energy consumption, and temperature. The objective of this study is to provide practical insights on the trade-offs among hardware options and to identify configurations that are suited for scalable, real-time rice disease detection in field conditions.

2. Methodology

2.1 Dataset description

Field surveys conducted at the Vietnam National University of Agriculture (VNUA) provided the source material for this research. Annotation correctness and consistency were verified in consultation with an expert from VNUA and the resulting dataset was manually inspected to discard low-quality images to ensure appropriate input quality for training. The final collection contains 3,542 images, resized to 320x320 pixels, covering three distinct diseases: leaf blast, brown spot, and leaf folder. The dataset was randomly partitioned into three subsets with 80% for training, 10% for validation, and 10% for testing. This split ensured that all disease types were present in each subset [7]. Table 1 demonstrates the distribution of the dataset images.

Given the limited size of the dataset, data augmentation was introduced to strengthen the model's ability to generalized and to limit overfitting [8]. The augmentation process was configured to emulate realistic variability encountered in field conditions such as illumination, viewpoint, and leaf pose. The applied transformations included random horizontal flipping, rotation, and scaling. These augmentation operations were performed stochastically during training, resulting in varied visual representations of the dataset across training epochs. Specifically, horizontal flipping was employed in the left-right direction with a probability of 50%, while rotation angles were randomly selected in the range of -20° to 20°. As for scaling, the input images were scaled up or down within 50% of their original size.

Table 1: Distribution of rice disease dataset

	Leaf blast	**Brown spot**	**Leaf folder**	**Total**
Training	919	923	987	2829
Validation	113	114	112	349
Testing	118	119	127	364
Total	**1150**	**1156**	**1236**	**3542**

2.2 YOLOv5n

The YOLOv5n (Nano) object detection model aligns well with the design requirements of the rice disease detection system due to its compact architecture and suitability for deployment across different processors [9]. Featuring roughly 1.9 million parameters and minimal computational demands, the nano variant supports real-time inference on resource-limited devices where both memory and processing power are constrained. While larger YOLO versions offer higher accuracy, YOLOv5n provides the necessary trade-off between detection performance, inference speed and resource utilization, rendering it the optimal candidate for field deployment in energy-constrained agricultural environments. Figure 1 provides an overview of the YOLOv5n architecture, including backbone, neck, and detection head [10].

The backbone employs a Cross Stage Partial (CSP) Darknet architecture optimized for feature extraction from rice leaf images. The CSP architecture reduces computational operations and model size while preserving gradient propagation. This design enables effective multi-scale feature extraction, capturing visual patterns ranging from fungal lesions in blast disease to large-area discolorations in brown spot.

The Path Aggregation Network (PANet) functions as the neck module, managing the wide range of spatial scales and facilitating bidirectional feature fusion. By integrating complex patterns from deeper layers with fine details from shallower layers, PANet creates robust multi-scale feature representations. This improves the model's ability to simultaneously detect both small, early-stage spots and large, severe infections within the same image.

The detection head performs parallel predictions on three distinct scales, concurrently generating bounding boxes, confidence scores and class probabilities for the detected diseases. Using high, medium and low-resolution feature maps, the model ensures consistent detection across varying object scales.

2.3 Optimization techniques

To enable deployment of the YOLOv5n rice disease detection model on resource-constrained edge platform, an optimization process converts the floating-point model into a compatible format for hardware.

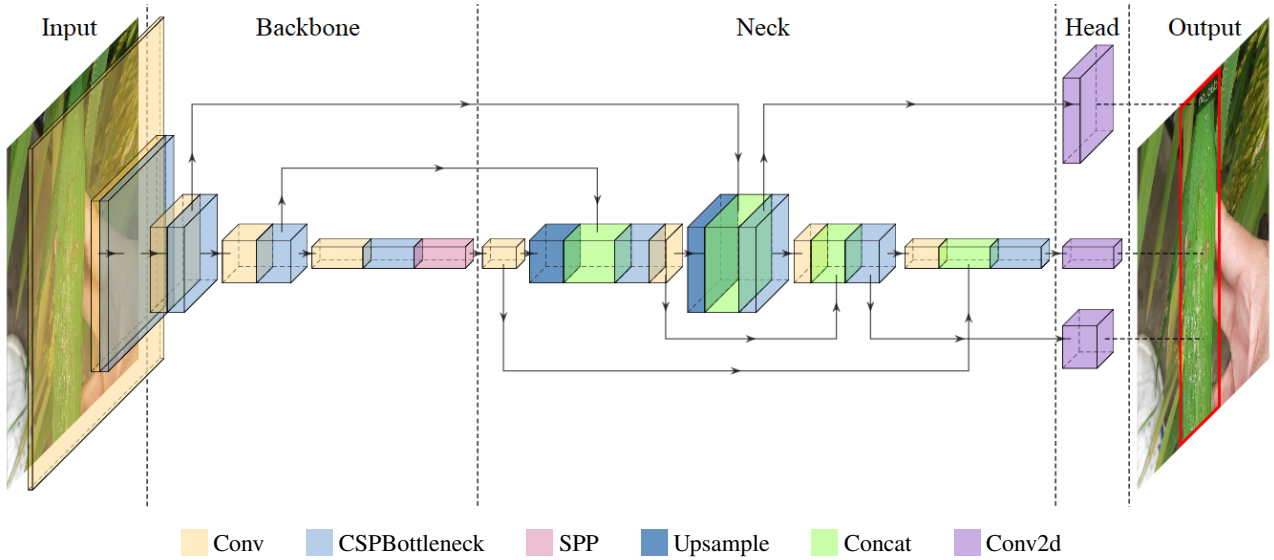


Figure 1: Architecture of the YOLOv5n framework for rice disease classification

After the training was completed, model quantization was applied to reduce computational and memory requirements of the trained YOLOv5n model as the native model utilizes FP32 precision, which exceeds memory usage and computational capacity. Quantization method converts the model's FP32 weights and activations to lower-precision INT8 format within the TensorFlow Lite framework, reducing model size and improving inference speed. Unlike FP32, which uses 32 bits to achieve a broad range of continuous values, INT8 employs 8-bit integers for a four times reduction in memory usage. This replaces intensive floating-point operations with integer arithmetic suited for edge processors. A small calibration dataset of rice field images helps set optimal scale factors for each layer, minimizing quantization error while keeping detection accuracy for rice diseases [6]. This post-training approach maintains the original network architecture unchanged and produces a model optimized for embedded platforms.

2.4 Evaluation metrics

The rice disease detection system's performance was evaluated using standard object detection metrics, specifically Precision, Recall, F1-score, and mean Average Precision (mAP). Precision measures the proportion of correctly identified disease regions among all detections, whereas recall assesses the model's sensitivity in identifying all actual disease regions. F1-score serves as the harmonic mean of precision and recall, providing a balanced assessment of classification performance. Furthermore, the accuracy of the bounding box positioning was quantified mAP_{50} estimates moderate bounding box overlap at Intersection over Union (IoU) at 0.5 and $\text{mAP}_{50:95}$ across IoU thresholds 0.5 to 0.95. These metrics determine overall performance across all object classes. The formulas [11] are:

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

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{2}$$

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

$$\text{AP} = \int_0^1 P(R)\mathrm{d}R \tag{4}$$

$$\text{mAP} = \frac{1}{C}\sum_{i=1}^{C} \text{AP}_i \tag{5}$$

where C denotes the total number of classes and AP_i represents the average precision calculated for the i-th class.

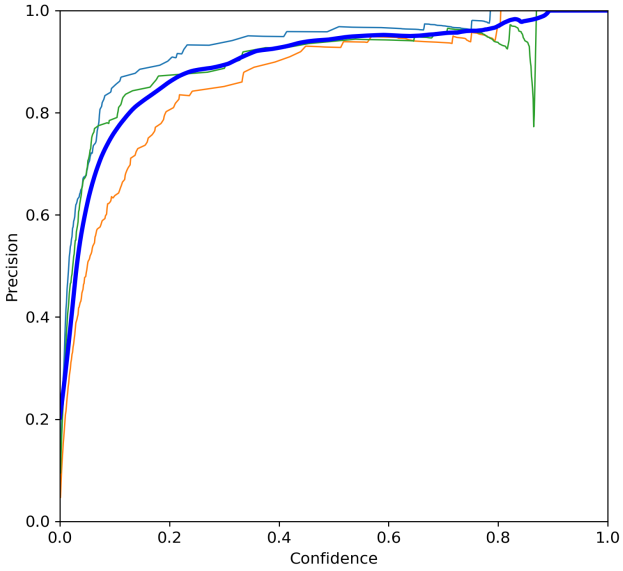
The operational efficiency across the hardware configurations is reviewed by measuring inference speed, device temperature, power consumption and CPU utilization to signify the trade-off between computational throughput and resource costs.

3. Deployment & experiments

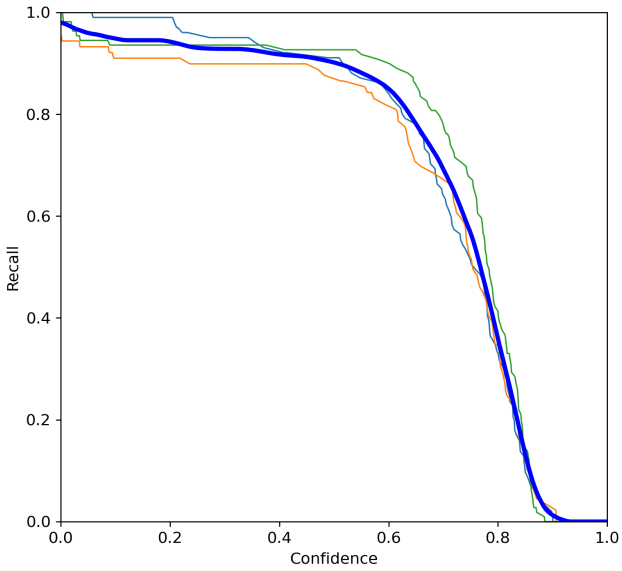
3.1 Rice disease detection model

The quantized YOLOv5n model was deployed for rice leaf disease detection and classification from field-captured images. On the test dataset, the model exhibited stable detection performance, as shown in Table 3. Most disease classes were correctly identified. Model detection characteristics across varying confidence thresholds appear in the F1-score, Precision, Precision-Recall, and Recall curves in Fig. 2a through Fig. 2d. The curves exhibit smooth and stable trends, indicating that the balance between precision and recall persisted after quantization. This confirms that the reduction in numerical precision did not substantially degrade model inference behavior.

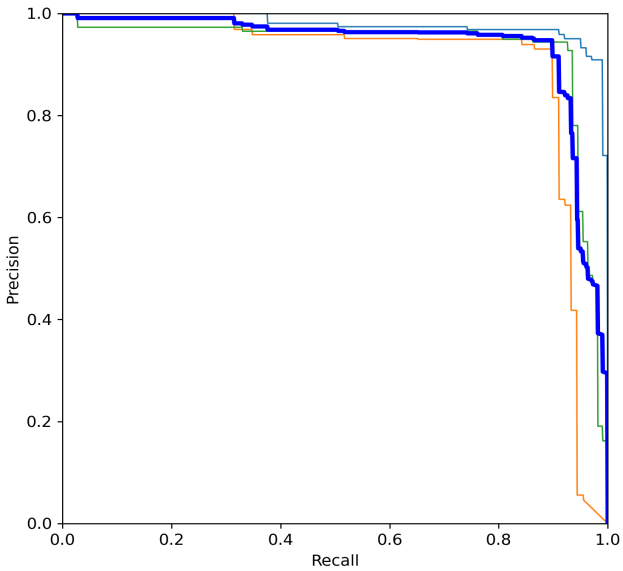
The overall Precision and Recall values indicate that most detected disease regions are correct and that the majority of actual disease instances are successfully identified. This balance is reflected by an overall F1-score, indicating stable detection behavior without a pronounced tendency toward either false positives or missed detections. At the class level, Leaf Blast attains the highest accuracy, indicating that the visual characteristics of this disease are effectively learned by the model. Brown Spot exhibits a lower Recall score, suggesting



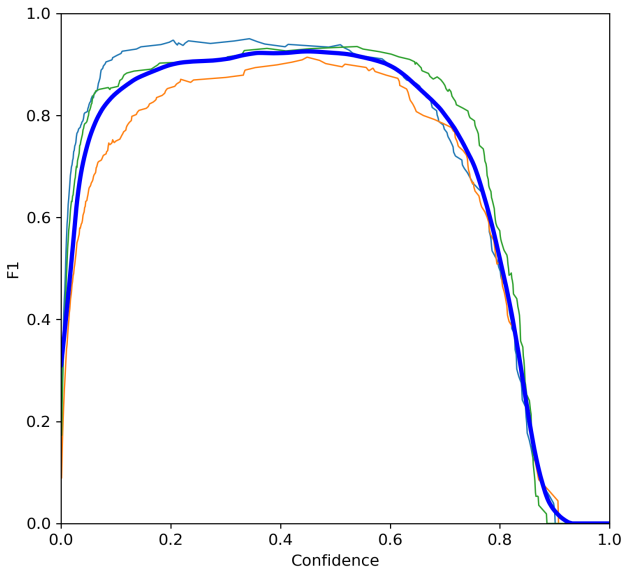
(a) Precision-Confidence curve



(b) Recall-Confidence curve



(c) Precision-Recall curve



(d) F1-Confidence curve

Figure 2: Performance metrics of the YOLOv5n model for rice disease detection

Table 2: The primary specifications of the experimental hardware platforms

	Workstation	**Raspberry Pi 4 Model B**	**PSoC Edge E84 AI Kit**
Processor	Intel Core i7-13700K (16-Core)	Broadcom BCM2711 (4-Core)	PSE846GPS2DBZC4 (2-Core)
	8P + 8E Cores	Cortex-A72	Cortex-M33 + Cortex-M55
Clock freq.	5.4 GHz	1.5 GHz	400 MHz
Memory	64 GB DDR4	8 GB LPDDR4	16 MB PSRAM
Accelerator	Intel UHD Graphics 770	N/A	Arm Ethos-U55 NPU + Helium DSP

that some affected regions remain undetected; however, the high Precision indicates limited confusion with other classes. For Leaf Folder, the Recall reaches a similarly high level, indicating good coverage of disease regions, while the lower $mAP_{50:95}$ value reflects reduced bounding box localization accuracy under stricter IoU thresholds, which may be related to the elongated shape and indistinct boundaries of this disease type. Although mAP_{50} remains high across all classes,

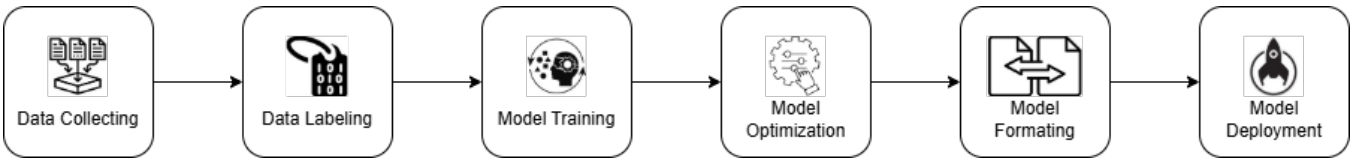


Figure 3: Deployment workflow of the proposed rice disease detection system

Table 3: Evaluation of the quantized model on test set

	P	R	mAP_{50}	$mAP_{50:95}$	F1-score
Leaf Blast	0.958	0.914	0.976	0.635	0.935
Brown Spot	0.93	0.892	0.902	0.635	0.911
Leaf folder	0.936	0.927	0.934	0.522	0.931
All	**0.941**	**0.911**	**0.937**	**0.544**	**0.926**

a notable decrease is observed for $mAP_{50:95}$, reflecting the increased difficulty of precise localization under higher IoU requirements.

3.2 Deployment pipeline

To bridge the operational gap between high-performance training environments and the resource-constrained hardware, a six-stage implementation pipeline is adopted, as illustrated in Fig. 3. This workflow is essential to operate within the strict memory and power limits of the target hardware.

The process begins with data collection and labeling, where field imagery is acquired and annotated to define the correct class labels for the three target disease classes. Model training is performed on a high-performance workstation using the PyTorch framework to obtain the initial model weights and evaluate its accuracy. To address the computational limitations of the embedded platform, the model optimization phase applies post-training quantization which reduces the memory footprint. Next, model formatting serializes the quantized network into C-array header and source files, rendering it compatible with the device firmware. Finally, model deployment integrates these components into the target hardware platform, enabling standalone inference without external dependencies.

3.3 Hardware setup

In our study, we selected three hardware platforms that represent distinct classes of computing performance to evaluate the deployment feasibility of the proposed system. The selection spans the operational spectrum from high-performance development environments to power-constrained field devices. The specific platforms examined in this study include the Intel Core i7-13000K Workstation, the Raspberry Pi 4 Model B, and the PSoC Edge E84 AI Kit. Table 2 illustrates some of the key specifications for the hardware configurations.

The Intel Core i7-13700K, part of the 13th Generation "Raptor Lake" family, serves as the development workstation. It features a performance hybrid architecture with 16 physical cores (8 Performance-cores and 8 Efficiency-cores) and 24 logical threads, capable of reaching clock speeds up to 5.4 GHz. This platform is also supported by Intel UHD Graphics 770 and Intel Deep Learning Boost instructions. While it offers maximum computational throughput, its power consumption restricts its use to non-portable applications.

Raspberry Pi 4B, serving as the intermediate platform, is one of the most widely adopted embedded systems for general-purpose computing. It is powered by a 64-bit Broadcom BCM2711 quad-core Cortex-A72 System on Chip (SoC) running at 1.5 GHz and is equipped with flexible connectivity options. In this study, the RPi 4 represents a standard embedded computer that runs a full operating system. However, it relies on general-purpose CPU cores for inference tasks and lacks specialized neural acceleration hardware, resulting in different power and latency trade-offs.

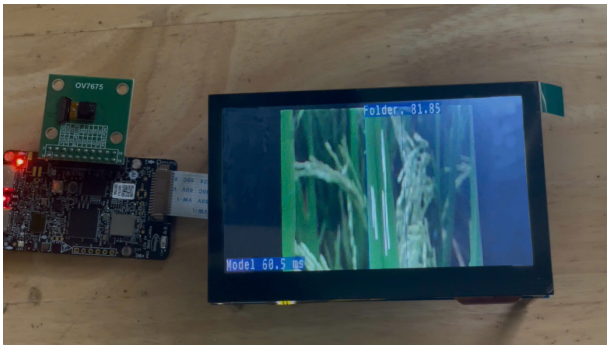


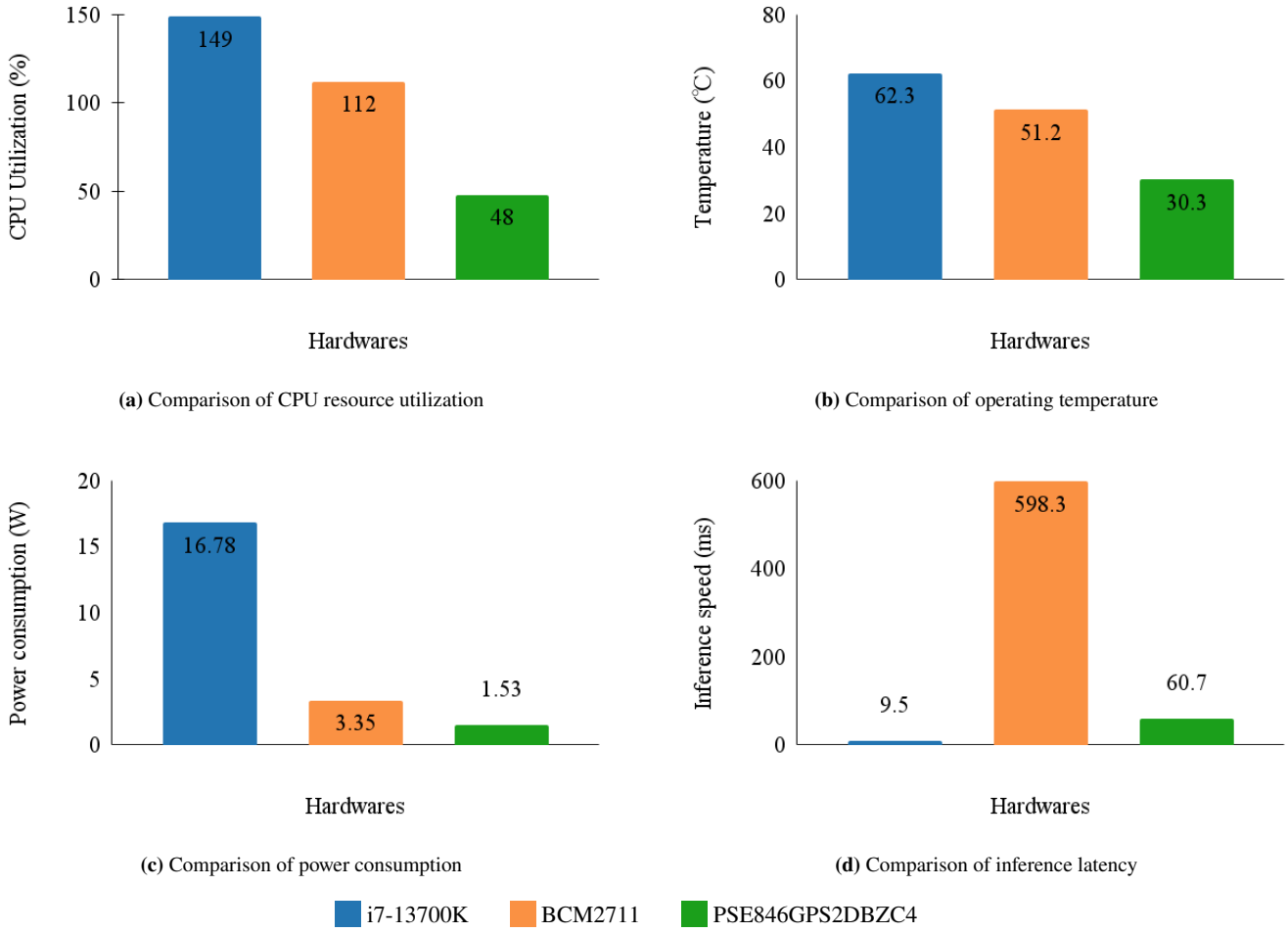
Figure 4: Output visualization

Infineon addresses the requirements of the target deployment environment through specialized embedded architecture. The PSoC Edge E84 AI Kit represents the ultra-low-power device in this study. This specialized computing platform integrates an Arm Cortex-M55 core for high-performance workloads and a secondary Cortex-M33 core for system management. Notably, it utilizes the Arm Ethos-U55 Neural Processing Unit, which accelerates quantized neural network operations. The integrated processing unit handles complex inference at very low power draw, rendering it the preferred platform for battery constrained field deployments. Figure 4 illustrates the proposed system performs real-time inference on the embedded device.

These hardware differences imply inevitable trade-offs between computational capability, system complexity, and energy consumption. Resource-rich platforms enable more flexible development and evaluation but require larger energy and hardware infrastructure. Conversely, resource-constrained embedded platforms permit continuous operation under limited energy conditions at the cost of reduced processing capacity.

3.4 Performance evaluations

Figure 5a shows average CPU utilization rates, normalized by core count for each platform, with the Intel i7 13700K having 24 logical cores, the Raspberry Pi 4 having 4 cores, and the PSoC Edge E84 having 2 cores but relying on the Cortex-M55 core alone for high performance inference tasks. The high per-



(a) Comparison of CPU resource utilization

(b) Comparison of operating temperature

(c) Comparison of power consumption

(d) Comparison of inference latency

Figure 5: Hardware performance

formance workstation (with Intel Core i7-13700K) recorded a value of 149%, given the processor's 24 logical threads, this is equivalent to 6.2% per logical core. This indicates that the inference workload is negligible for the workstation, leaving the vast majority of its computational resources and 64 GB of system memory available for other tasks. Similarly, the Raspberry Pi 4 recorded a utilization of 112%, reflecting utilization beyond one core but below full capacity across its four cores (approximately 28% total system load). This demonstrates moderate resource requirements that prevent system saturation under the given model demands. The PSoC Edge E84 maintained 48% utilization on the Cortex M55 because the Arm Ethos-U55 NPU handles tensor operations.

Figure 5b presents thermal performance, reflecting workload distribution and hardware characteristics during extended inference sessions. The desktop processor reached the highest average temperature of 62.3°C, followed by the RPi 4 at 51.2°C. Despite operating below its total system capacity, the thermal concentration on the active core resulted in elevated temperatures. Conversely, the specialized edge kit maintained a near-ambient operating temperature of 30.3°C, indicating its suitability for enclosed, fanless deployments in agricultural settings.

Figure 5c measures power consumption, a critical parameter for hardware platforms where power efficiency determines practical deployment limits. The CPU, serving as the high-performance baseline, consumed 16.78 W, while the RPi 4 operated at 3.35 W. While it offers a reduction in power compared to the workstation, the microcontroller-based platform demonstrated superior energy efficiency, consuming only 1.53 W during active inference. This represents a reduction in power consumption of approximately 54% compared to the Raspberry Pi 4.

Regarding inference speed, shown in Fig. 5d, the Intel i7-13700K naturally achieved the lowest latency of 9.5 ms due to its high clock speed and abundant resources, enabling rapid processing even at low thread utilization. However, a critical comparison arises between the two edge platforms. The RPi 4 recorded a significantly high latency of 598.3 ms (approximately 1.6 FPS). When viewed alongside its utilization data (112%), this high latency confirms that the Cortex-A72 cores struggle to process the model efficiently in a sequential manner. In contrast, the PSoC Edge E84 achieved an inference speed of 60.7 ms (approximately 16 FPS). Despite having a lower clock speed than the RPi 4, the PSoC E84 performed nearly 10 times faster, highlighting the advantage of NPU acceleration over general-purpose CPU execution.

The evaluation results indicate clear performance–energy trade-offs across the three examined hardware platform categories. High-performance CPUs offer the lowest latency, making them ideal for development and validation, yet their power and thermal demands limit field utility. General-purpose embedded platforms, occupying in an intermediate position and offering improved energy efficiency compared with desktop

CPUs. Nevertheless, inference latency on these platforms is substantially higher, and their moderate power consumption constraints sustained real-time inference under strict latency requirements. Low-power edge devices prioritize energy efficiency by maintaining stable thermal conditions and minimal power consumption suitable for battery-powered systems. Although inference latency remains higher than high-performance platforms, the achieved throughput is well within the operational parameters required for autonomous surveillance applications.

3.5 Discussion

Prior YOLO-based plant disease detection approaches have largely focused on maximizing accuracy by employing enhanced models on high-performance computing platforms. For instance, Gao et al. introduced a "YOLOv5-Efficient" variant (based on YOLOv5s) that improved average precision by roughly 9.9% over the original YOLOv5, but this gain was achieved using a GPU for inference. Similarly, Cheng et al. developed an improved YOLOv7-Tiny model to boost detection accuracy (reaching an mAP_{50} of 92.2%) at the cost of a larger network (approximately 12.2 million parameters) running on relatively powerful hardware. These works demonstrate that advanced modifications of YOLO can yield high detection accuracy, but their evaluations assume ample computational resources and do not address deployment constraints such as energy efficiency or device autonomy.

In contrast, this study pursues a complementary goal which is practical real-time deployment on an extremely resource-constrained platform. We quantize and deploy YOLOv5n on an Infineon PSoC Edge E84 AI Kit equipped with an NPU, focusing on feasibility of AI at the edge. This approach prioritizes low power consumption and stand-alone operation, the device can perform on-board inference without any external accelerator or cloud support. While not aiming to surpass or in accuracy, the proposed solution demonstrates that acceptable detection performance is achievable entirely on-device with small computational resources. This design trade-off highlights a different but complementary direction of research, emphasizing energy-efficient, autonomous operation for real-world agricultural monitoring.

4. Conclusion

Experimental results confirm a clear performance hierarchy among the three hardware platforms evaluated in this study. The high-end workstation delivered the fastest inference with minimal latency but with the cost of high power and resource usage. The Raspberry Pi 4 exhibited significantly slower inference speeds, making it more suitable as a prototyping intermediate platform rather than a final field deployment solution. In contrast, the low-power PSoC Edge E84 AI Kit microcontroller is equipped with an Arm Ethos-U55 NPU accelerator that achieved an inference time of approximately 60.7 ms per image (roughly 16 FPS) while ensuring stable operation under ambient conditions. Notably, the post-quantization YOLOv5n model maintained consistent disease detection accuracy across all platforms, indicating that the hardware constraints impacted throughput far more than predictive accuracy.

These findings demonstrate the feasibility of deploying advanced object detection models on microcontroller-class edge devices for real-time plant disease monitoring without compromising accuracy. Our comparative analysis provides practical guidance for edge AI system design: high-performance processors are ideal for rapid development and testing, whereas final deployments benefit from low-power platforms that satisfy field operational constraints. In IoT-based agricultural settings, battery-powered edge devices with on-board NPUs enable autonomous, continuous crop monitoring in remote areas without reliance on grid power. Therefore, system designers must balance computational performance with operational sustainability based on specific application requirements. In practice, factors such as energy availability, environmental conditions, and maintenance overhead often influence design decisions more strongly than achieving the absolute lowest inference latency.

Acknowledgement

The research is supported by the Industrial Instrumentation & IoT Laboratory at the School of Electrical and Electronic Engineering, Hanoi University of Science and Technology. The authors would like to thank the experts at the Vietnam National University of Agriculture for providing dataset and assistance.

References

[1] Maitah, K., Smutka, L., Sahatqija, J., Maitah, M., Anh, N. P. (2020). Rice as a Determinant of Vietnamese Economic Sustainability. Sustainability, 12(12), 5123.
[2] Devanna, A. S., Jain, P. K., Sharma, T. R. (2022). Understanding the Dynamics of Blast Resistance in Rice-Magnaporthe oryzae Interactions. Journal of Fungi, 8(6), 584.
[3] Pallathadka, H., Ravipati, P., Sajja, G. S., Phasinam, K., Kassanuk, T., Sanchez, D. T., Prabhu, P. (2022). Application of machine learning techniques in rice leaf disease detection. Materials Today: Proceedings, 51, 2277-2280.
[4] Gao, W., Zong, C., Wang, M., Zhang, H., Fang, Y. (2024). Intelligent identification of rice leaf disease based on YOLO V5-EFFICIENT. Crop Protection, 183, 106758.
[5] Cheng, D., Zhao, Z., Feng, J. (2024). Rice Diseases Identification Method Based on Improved YOLOv7-Tiny. Agriculture, 14(5), 709.
[6] Hubara, I., Nahshan, Y., Hanani, Y., Banner, R., Soudry, D. (2021). Accurate Post Training Quantization With Small Calibration Sets. 38th International Conference on Machine Learning.
[7] Muraina, I. O. (2021). Ideal dataset splitting ratios in machine learning algorithms: General concerns for data scientists and data analysts. 7th International Mardin Artuklu scientific researches conference, 496-504.
[8] Perez, L., Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621.
[9] Khanam, R., Hussain, M. (2024). What is YOLOv5: A deep look into the internal features of the popular object detector. arXiv preprint arXiv:2407.20892.
[10] Wang, C., Wang, C., Wang, L., Wang, J., Liao, J., Li, Y., Lan, Y. (2023). A Lightweight Cherry Tomato Maturity Real-Time Detection Algorithm Based on Improved YOLOV5n. Agronomy, 13(8), 2106.
[11] Wang, B., Yan, Y., Lan, Y., Wang, M., Bian, Z. (2023). Accurate Detection and Precision Spraying of Corn and Weeds Using the Improved YOLOv5 Model. IEEE Access, 11, 29868-29882.