Comparative Benchmarking of YOLO-Based Object Detection on Raspberry Pi 3B, 4B, and 5B Under CPU-Only Edge AI Deployment

Marcelo Tsuguio Okano, William Aparecido Celestino Lopes and Sergio Miele Ruggero

Abstract— **CPU-only Edge AI deployment on low-cost Single Board Computers requires more than ranking hardware by inference speed; it requires determining whether a model-runtime-hardware configuration satisfies application-specific feasibility constraints. This paper presents a feasibility-oriented benchmark of YOLOv8n object detection on Raspberry Pi 3B, Raspberry Pi 4B, and Raspberry Pi 5B under CPU-only execution using PyTorch and ONNX Runtime. The MVTec AD dataset, originally designed for anomaly detection, was adapted to a supervised object-detection setting by converting defective regions into bounding-box annotations and retaining non-defective images as negative samples, thereby enabling controlled evaluation of industrial visual inspection workloads. YOLOv8n was trained on a workstation equipped with an NVIDIA RTX 4060 GPU and exported to native PyTorch (.pt) and ONNX (.onnx) formats. Each configuration processed 493 images at 640 x 640 pixels after a warm-up phase, while latency, throughput, CPU usage, memory usage, and device temperature were monitored. To move beyond descriptive benchmarking, the results are interpreted through explicit operational feasibility classes that combine performance and thermal admissibility. The findings show that hardware generation strongly improves latency, but runtime choice and thermal behavior determine practical viability. ONNX Runtime increases CPU saturation and thermal load, whereas PyTorch provides a more balanced profile on the tested ARM-based SBCs. The Raspberry Pi 3B remains suitable only for offline inference, the Raspberry Pi 4B supports low-cadence inspection, and the Raspberry Pi 5B becomes computationally viable for low-rate edge inspection, provided that adequate cooling is used. The study contributes a reproducible and decision-oriented benchmarking protocol for selecting frugal SBC configurations for industrial Edge AI deployment.**

I. Introduction

Recent advances in Artificial Intelligence (AI) have driven substantial transformations in industrial systems, particularly through the integration of computer vision techniques into inspection, monitoring, and automation processes. In parallel, there has been a progressive shift from centralized computing architectures toward decentralized approaches, in which data processing is performed closer to the source. This transition aims to reduce latency, dependency on network connectivity, and operational costs, and it underpins the paradigms commonly referred to as the Internet of Things (IoT), Edge AI, and TinyML.

Within this context, low-cost and energy-efficient platforms such as Single Board Computers (SBCs) have emerged as key enablers for Edge AI applications. Among these platforms, the Raspberry Pi family stands out due to its widespread adoption in both academic research and industrial prototyping, often serving as a reference hardware platform for embedded and edge-based systems. Despite its popularity, however, there remains uncertainty regarding the practical capability of these devices to execute modern computer vision models efficiently and reliably when restricted to CPU-only execution.

At the same time, object detection architectures based on deep learning have evolved rapidly, making it feasible to deploy increasingly complex vision tasks on resource-constrained platforms. Nevertheless, practical deployment on SBCs involves important trade-offs among computational performance, resource utilization, and thermal behavior, which vary significantly depending on hardware generation and execution environment.

Moreover, evaluations based solely on isolated performance metrics, such as average inference latency or frames per second (FPS), are insufficient to characterize the feasibility of Edge AI systems in real-world scenarios. In industrial and embedded contexts, system-level factors including CPU and memory utilization, thermal stability, and sustained performance under continuous workloads are equally critical and are often overlooked.

Motivated by these considerations, this paper aims to systematically evaluate the behavior of Single Board Computers from the Raspberry Pi family, specifically the Raspberry Pi 3 Model B, Raspberry Pi 4, and Raspberry Pi 5, when executing a YOLO-based object detection model exclusively on CPU, without the use of GPUs or dedicated hardware accelerators. The experimental analysis considers two widely adopted execution environments, native PyTorch inference (.pt) and ONNX Runtime inference (.onnx), and employs an integrated set of metrics including inference latency, effective throughput, CPU and memory utilization, and device temperature.

M.T.O. is with the Graduate Program in Production Engineering Paulista University UNIP, SP 04026-002 Brazil (corresponding author phone: 55-11-991096575555-5555; e-mail: marcelo.okano1@docente.unip.br).

W.A.C.L. is with the Graduate Program in Production Engineering Paulista University UNIP, SP 04026-002 Brazil (e-mail: wilnatelha@gmail.com).

S.M.R. is with the Graduate Program in Production Engineering Paulista University UNIP, SP 04026-002 Brazil (e-mail: smruggero@uol.com.br).

To avoid treating the experiment as a simple hardware ranking, this study formulates CPU-only Edge AI deployment as a feasibility decision problem. Let $C = \{c_i\}$ denote the set of candidate configurations, where each configuration combines a hardware platform, an inference runtime, and a fixed detection model. For each configuration, the measured vector $m_i = \{L_i, F_i, U_i, M_i, T_i\}$ represents average latency, effective throughput, CPU utilization, memory usage, and operating temperature. A configuration is considered operationally feasible only if its measured behavior satisfies application-dependent admissibility conditions for throughput, latency, resource use, and thermal stability. Therefore, the objective is not merely to identify the fastest Raspberry Pi generation, but to determine which configurations remain admissible for different classes of low-cost industrial inspection tasks under CPU-only constraints.

Accordingly, this paper makes three contributions. First, it proposes a feasibility-oriented benchmark for CPU-only Edge AI deployment on low-cost SBCs, integrating performance, resource consumption, and thermal indicators into a single decision perspective. Second, it details a reproducible experimental protocol for adapting an industrial anomaly-detection dataset to supervised object detection and for executing the same YOLOv8n model across Raspberry Pi 3B, 4B, and 5B platforms. Third, it provides deployment-oriented evidence showing that runtime selection may change the operational classification of a configuration, since high CPU utilization and thermal load can offset nominal throughput gains.

II. RELATED WORKS

The deployment of deep learning–based object detection at the network edge has been widely investigated as a response to the increasing demand for low-latency visual intelligence in industrial, agricultural, and cyber-physical systems. Within this context, the YOLOv8 family of one-stage detectors has emerged as a dominant solution due to its unified detection pipeline and comparatively low computational overhead, making it particularly suitable for execution on resource-constrained platforms [1]–[3]. Prior studies consistently demonstrate that YOLO-based models provide a favorable balance between detection accuracy and inference efficiency when compared to more complex two-stage architectures, especially in scenarios where real-time or near-real-time response is required.

YOLO (You Only Look Once) refers to a family of single-stage object detectors that perform object localization and classification in a single forward pass through the neural network. Unlike two-stage detectors, which first generate region proposals and then classify them, YOLO directly predicts bounding boxes and class probabilities from the input image. This unified detection pipeline reduces inference latency and makes YOLO-based models attractive for Edge AI applications, particularly when deployment is restricted to resource-constrained processors such as the CPUs of Single Board Computers.

The applicability of YOLO architectures to edge scenarios has been validated across multiple domains. In agricultural environments, YOLO models have been successfully employed for fruit detection and classification under varying illumination and scale conditions [1]. In industrial contexts, YOLO-based approaches have been applied to quality inspection and visual conformity assessment, highlighting their robustness in structured production settings [2], [3]. Similarly, ecological monitoring applications further confirm the adaptability of YOLO detectors to diverse visual patterns and constrained hardware environments [4]. These works collectively establish YOLO as a de facto standard for edge-oriented object detection.

From a hardware perspective, Single Board Computers (SBCs) have received significant attention as deployment platforms for Edge AI solutions due to their low cost, compact form factor, and ease of integration. Among SBCs, the Raspberry Pi family has been one of the most extensively evaluated platforms in the literature [2], [3]. Early investigations consistently report that the Raspberry Pi 3 Model B is limited to offline inference, primarily as a consequence of restricted CPU throughput and memory bandwidth [5], [6]. These limitations confine its use to proof-of-concept implementations or low-frequency processing tasks.

Subsequent generations have progressively expanded the feasibility of SBC-based inference. Studies focusing on the Raspberry Pi 4 Model B report substantial performance improvements, enabling near-real-time execution for lightweight deep learning models under controlled workloads [4], [13]. However, these gains are often accompanied by increased power consumption and thermal stress, which may limit sustained operation. More recent reports indicate that the Raspberry Pi 5 further improves computational capability, extending the range of viable workloads, while simultaneously introducing stricter requirements for thermal management and power delivery [2]. This evolution highlights that hardware advances alone do not fully resolve the challenges associated with long-term Edge AI deployment.

In addition to hardware considerations, the choice of inference runtime has been identified as a critical factor influencing performance and system stability. Native PyTorch inference remains prevalent in experimental and prototyping scenarios due to its robustness and seamless integration with training pipelines [7], [9]. Conversely, ONNX Runtime has been proposed as a framework-agnostic alternative that facilitates model portability across heterogeneous platforms. Despite its interoperability advantages, prior studies report that ONNX-based inference may exhibit performance variability, conversion overhead, and suboptimal optimization on resource-constrained edge devices [8], [10].

Recent benchmarking efforts argue that evaluating Edge AI systems solely based on isolated metrics such as inference latency or frames per second provides an incomplete assessment of real-world feasibility [6], [11]. Instead, comprehensive evaluations that jointly analyze inference performance, CPU utilization, memory consumption, and thermal behavior are required to capture the operational constraints of SBC-based deployments [12]. Furthermore, studies on thermal-aware and energy-aware execution demonstrate that sustained inference must explicitly consider temperature and power dynamics to avoid throttling, instability, or premature hardware degradation [14]–[16]. Emerging collaborative and distributed inference paradigms further reinforce the need for holistic benchmarking

methodologies that integrate both computational performance and system-level constraints [17].

III. Methodology

A. Research Design and Feasibility Formulation

This study was designed as a controlled feasibility-oriented benchmark for CPU-only Edge AI deployment on low-cost Single Board Computers. The experimental unit is a deployment configuration composed of one hardware platform, one inference runtime, and one fixed object-detection model. The candidate hardware platforms were Raspberry Pi 3 Model B, Raspberry Pi 4 Model B, and Raspberry Pi 5 Model B. The evaluated runtimes were native PyTorch inference and ONNX Runtime inference. For each configuration, latency, effective throughput, CPU utilization, memory usage, and operating temperature were collected. The central research question was: under which runtime and hardware conditions does a lightweight YOLO-based detector remain operationally feasible for low-cost industrial inspection without GPU, NPU, or external accelerator support?

B. Dataset Adaptation from MVTec AD to Object Detection

The MVTec AD dataset was selected because it contains industrial object and surface categories representative of visual inspection scenarios. However, MVTec AD was originally designed for anomaly detection rather than supervised object detection. Therefore, an explicit conversion procedure was required. In the revised dataset, defective samples were converted into object-detection annotations by deriving bounding boxes from the available defect regions/masks. For each defective region, the smallest enclosing rectangle was computed and exported in YOLO annotation format. Non-defective images were retained as negative samples with empty label files, allowing the detector to be evaluated in the presence of normal industrial images. The detection task was formulated as single-class defect detection. The dataset was organized using an 80/10/10 train/validation/test split, and the same 493-image test subset was used for all Raspberry Pi experiments.

C. Annotation and Class Definition

Annotations were generated according to the YOLO format, where each label file contains the class identifier and normalized bounding-box coordinates. A single defect class was adopted because the objective of the study was not to maximize category-level defect classification accuracy, but to create a consistent industrial inspection workload for evaluating CPU-only inference feasibility across SBC generations and runtimes. All images were resized to 640 x 640 pixels for training and inference, ensuring comparability across platforms.

D. Model and Training Procedure

YOLOv8n was selected because it is the smallest member of the YOLOv8 family and is therefore suitable for CPU-only deployment on resource-constrained SBCs. Training was performed on a workstation equipped with Ubuntu 24.04 LTS, 16 GB RAM, and an NVIDIA GeForce RTX 4060 GPU using the Ultralytics implementation integrated with PyTorch. The parameters are in Table I.

TABLE I. Training Parameters and Experimental Configuration Used in the Revised Protocol.

Parameter	Adopted value in the revised protocol
Model	YOLOv8n
Input size	640 x 640 pixels
Dataset	MVTec AD converted to YOLO object-detection format
Detection class	Single class: defect
Train/validation/test split	80/10/10
Test subset used on Raspberry Pi devices	493 images
Epochs	100
Batch size	16
Optimizer	AdamW
Initial learning rate	0.001
Early stopping patience	30 epochs
Pretrained weights	YOLOv8n pretrained weights
Random seed	42
Augmentation	Ultralytics standard augmentation: scale, translation, horizontal flip, HSV adjustment, and mosaic
Model selection criterion	Best validation mAP@0.5:0.95
Confidence threshold	0.25
IoU/NMS threshold	0.45
Inference batch size	1
Training hardware	NVIDIA GeForce RTX 4060 GPU, 16 GB RAM, Ubuntu 24.04 LTS
Inference hardware	Raspberry Pi 3B, Raspberry Pi 4B, Raspberry Pi 5B
Inference runtimes	PyTorch/Ultralytics and ONNX Runtime

E. Model Export and Runtime Environments

After training, the selected model weights were exported to two deployment formats: native PyTorch (.pt), executed through the Ultralytics inference pipeline, and ONNX (.onnx), executed through ONNX Runtime. The same trained weights, image resolution, confidence threshold of 0.25, IoU/NMS threshold of 0.45, and inference batch size of 1 were used across all experiments. This design isolates the influence of hardware generation and runtime environment on CPU-only deployment behavior.

F. Inference Protocol on Raspberry Pi Platforms

Inference was executed on Raspberry Pi 3B, Raspberry Pi 4B, and Raspberry Pi 5B under CPU-only conditions, without GPUs, NPUs, or external accelerators. Each experiment began with 20 warm-up iterations, which were excluded from the reported metrics. The test set of 493 images was then processed sequentially at 640 x 640 pixels. Temperature was treated as an

accessible operational thermal indicator. Direct power consumption measurement and longer thermal stress tests are considered future extensions.

G. Metrics and Statistical Treatment

For each configuration, the following metrics were collected: average latency per image, effective frames per second, average CPU utilization, average RAM usage, and final operating temperature. The reported values provide controlled empirical evidence for the tested configurations. Since repeated-run confidence intervals were not available for all experiments, the manuscript avoids claims of statistical significance and focuses on feasibility-oriented comparison.

H. Reproducibility Package

To address reproducibility, the revised protocol specifies the dataset conversion logic, YOLO configuration, training command parameters, model export formats, inference thresholds, warm-up procedure, fixed test subset, hardware platforms, and the metrics collected on each Raspberry Pi device.

I. Threats to Validity

The main threats to validity are the use of a single YOLO variant, a fixed input resolution, a single industrial dataset, and temperature as a short-run operational indicator instead of direct power measurement. Therefore, the conclusions should not be generalized to all Edge AI models or all SBCs. Instead, the study provides a transferable benchmarking protocol and empirical evidence for the tested model-runtime-hardware configurations.

IV. RESULTS

The results are analyzed according to operational feasibility classes rather than as a simple ranking of Raspberry Pi generations. Configurations below 0.5 FPS are classified as offline-only, configurations between 0.5 and 1.0 FPS are classified as low-cadence, and configurations above 1.0 FPS are classified as near-online for low-rate inspection tasks. Thermal admissibility is evaluated separately because a configuration may be computationally viable but thermally unsuitable for unattended deployment without active cooling. This classification supports transparent hardware-runtime selection for CPU-only Edge AI deployment.

This section presents a comprehensive analysis of the experimental results obtained from executing a YOLO-based object detection model on three Single Board Computers: the Raspberry Pi 3 Model B (Pi 3B), Raspberry Pi 4 Model B (Pi 4B), and Raspberry Pi 5 Model B (Pi 5B), under two inference environments, namely PyTorch (.pt) and ONNX Runtime (.onnx). All experiments were conducted under strictly controlled conditions, using the same dataset, resolution, and execution protocol described in the Methodology section, ensuring full comparability across platforms and runtimes. The analysis integrates inference performance with system-level indicators, including CPU utilization, memory consumption, and thermal response, which are critical for assessing the feasibility of CPU-only Edge AI deployments.

A fixed dataset of 493 images was processed sequentially in all experiments, with an input resolution of 640×640 pixels and an initial warm-up phase of 20 iterations excluded from the analysis to mitigate transient initialization effects. Inference was performed exclusively on CPU, without GPUs or external accelerators, reflecting realistic low-cost industrial deployment scenarios. The evaluated metrics include average inference latency per image, effective frames per second (FPS), average CPU utilization, average RAM usage, and final device temperature.

The consolidated inference performance and resource utilization results for both PyTorch and ONNX runtimes are reported in Table II. For all platforms, PyTorch inference consistently achieves lower absolute latency, while ONNX Runtime exhibits significantly higher CPU utilization and thermal load. The Raspberry Pi 3B operates strictly in an offline regime under both runtimes, whereas the Raspberry Pi 4B enables asynchronous or low-cadence inference. The Raspberry Pi 5B achieves near–real-time performance, particularly in the PyTorch environment, albeit at the cost of elevated operating temperatures that necessitate active cooling.

Table II summarizes the experimental results obtained on the Raspberry Pi platforms. The observed CPU utilization and memory consumption patterns are directly related to the architectural characteristics of the YOLOv8n model, which prioritizes a reduced parameter count and lightweight convolutional blocks to enable execution on CPU-only edge platforms.

TABLE II. CONSOLIDATED INFERENCE PERFORMANCE AND RESOURCE UTILIZATION FOR PYTORCH (.PT) AND ONNX (.ONNX) RUNTIMES.

Platform	Runtime	Avg. Latency (ms)	FPS	Avg. CPU (%)	Avg. RAM (%)
Pi 3B	PyTorch	4304	0.23	40	51
Pi 3B	ONNX	5633	0.17	98	43
Pi 4B	PyTorch	1315	0.74	43	17
Pi 4B	ONNX	2023	0.48	98	15
Pi 5B	PyTorch	493	1.93	44	10
Pi 5B	ONNX	637	1.48	96	9

Figure 1 highlights the runtime-dependent thermal behavior, showing that ONNX Runtime systematically increases operating temperature across all platforms.

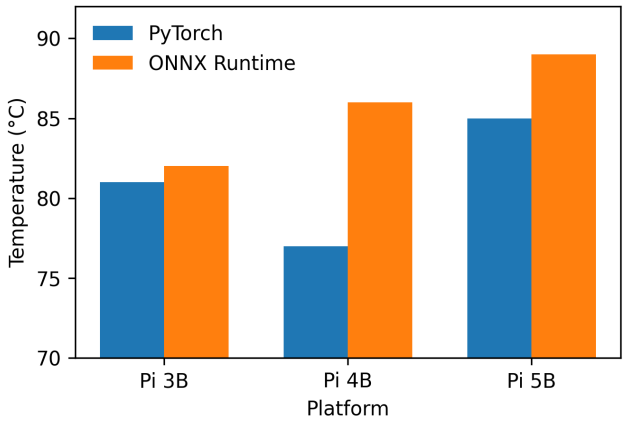

Fig. 1. Average operating temperature measured during sustained inference on Raspberry Pi platforms using PyTorch and ONNX Runtime. ONNX Runtime consistently increases thermal load, reaching 89 °C on the Raspberry Pi 5B, compared with 86 °C on the Pi 4B and 82 °C on the Pi 3B, reinforcing the thermal risk discussed for CPU-only Edge AI deployment.

To further quantify cross-generational scalability, derived metrics such as speedup and percentage performance gain are summarized in Table III, combining results from both inference environments. The data confirm that hardware evolution is the dominant factor driving performance improvements, with the Raspberry Pi 5B delivering nearly an order-of-magnitude speedup relative to the Raspberry Pi 3B, regardless of the runtime.

TABLE III. CROSS-GENERATIONAL SPEEDUP AND PERFORMANCE GAIN FOR PYTORCH AND ONNX INFERENCE.

Runtime	Comparison	Speedup	Performance Gain (%)
PyTorch	Pi 4B / Pi 3B	3.3×	69.5
PyTorch	Pi 5B / Pi 3B	8.7×	88.5
PyTorch	Pi 5B / Pi 4B	2.7×	63.0
ONNX	Pi 4B / Pi 3B	2.8×	64.1
ONNX	Pi 5B / Pi 3B	8.8×	88.7
ONNX	Pi 5B / Pi 4B	3.2×	68.5

Overall, the results show that the Raspberry Pi 5B achieves the highest throughput among the evaluated platforms and becomes suitable for low-rate near-online inspection tasks under CPU-only execution. However, its elevated operating temperature indicates that active cooling is required before considering unattended or long-duration industrial deployment. The Raspberry Pi 3B remains restricted to offline inference, while the Raspberry Pi 4B supports low-cadence inspection tasks. ONNX Runtime increases CPU utilization and thermal load across the tested platforms and does not consistently reduce latency compared with native PyTorch inference. Therefore, runtime selection affects not only speed but also operational admissibility.

V. DISCUSSION

The results should not be interpreted merely as a generational ranking of Raspberry Pi boards. The main analytical contribution is the operational distinction between performance and feasibility. In CPU-only Edge AI deployment, the fastest configuration is not necessarily the most admissible configuration if it saturates CPU resources, approaches thermal limits, or requires cooling conditions that are incompatible with the intended installation environment. This distinction is particularly relevant for frugal industrial deployments, where acquisition cost, maintainability, energy constraints, and environmental robustness may be as important as inference latency.

The comparison between PyTorch and ONNX Runtime reinforces this point. ONNX Runtime is often adopted because of its portability and framework-independent deployment model. However, in the tested ARM-based SBCs, ONNX Runtime increased CPU utilization and thermal load without consistently improving latency. This result suggests that runtime portability should not be treated as equivalent to operational superiority. For resource-constrained edge devices, the interaction among model architecture, runtime scheduling, CPU load, and thermal accumulation must be evaluated jointly.

The proposed feasibility classification also provides a practical decision aid. Raspberry Pi 3B configurations are restricted to offline inference and proof-of-concept use. Raspberry Pi 4B with PyTorch supports low-cadence inspection, such as periodic sampling or asynchronous quality-control checks. Raspberry Pi 5B supports low-rate near-online inspection, but only if thermal management is incorporated into the deployment design. Therefore, the study contributes a transferable benchmarking logic that can be reused to evaluate other SBCs, runtimes, and lightweight models.

This study also has limitations. First, it evaluates one lightweight YOLO variant at a fixed resolution, and the results should not be generalized to larger detectors or different computer vision tasks without additional experiments. Second, the benchmark focuses on PyTorch and ONNX Runtime; comparisons with TensorFlow Lite, quantized models, MobileNet-based detectors, or hardware accelerators remain necessary for a broader deployment map. Third, temperature was used as an accessible operational indicator, but direct power consumption measurements would provide a more precise characterization of energy efficiency. Finally, the adaptation of MVTec AD to object detection provides a controlled industrial workload, but future validation should include images collected from real production environments.

VI. CONCLUSION

This work presented a feasibility-oriented benchmark of YOLOv8n object detection on Raspberry Pi 3B, Raspberry Pi 4B, and Raspberry Pi 5B under CPU-only execution. Rather than treating FPS as the sole decision criterion, the study evaluated each configuration using latency, throughput, CPU utilization, memory usage, and operating temperature. The results show that Raspberry Pi 5B offers the best computational performance among the evaluated platforms, but also that runtime choice and thermal behavior strongly influence deployment admissibility.

Within the tested conditions, Raspberry Pi 3B is restricted to offline inference, Raspberry Pi 4B supports low-cadence inspection tasks, and Raspberry Pi 5B can support low-rate near-online inspection when adequate cooling is provided. The comparison between PyTorch and ONNX Runtime indicates that framework-independent deployment does not automatically improve operational feasibility on ARM-based SBCs, since higher CPU utilization may increase thermal stress and reduce sustainability under continuous workloads.

The main contribution of this study is a reproducible and feasibility-oriented benchmarking protocol for evaluating CPU-only Edge AI deployment on low-cost SBCs. Future work will extend the benchmark with direct power measurements, longer thermal stress tests, repeated runs with confidence intervals, TensorFlow Lite and quantized models, MobileNet-based baselines, and hardware accelerators. Additional experiments in real industrial inspection environments will also be conducted to validate the operational classes proposed in this work.

REFERENCES

[1] K. Buczyński, M. Kapłan, and Z. Jarosz, "Detection of Red, Yellow, and Purple Raspberry Fruits Using YOLO Models," Agriculture, vol. 15, no. 24, p. 2530, 2025.

[2] M. T. Okano et al., "Edge AI for Industrial Visual Inspection: YOLOv8-Based Visual Conformity Detection Using Raspberry Pi," Algorithms, vol. 18, no. 8, p. 510, 2025.

[3] W. Nugroho et al., "Automated Component Detection for Quality PCB Using YOLO Algorithm with IoT Real-Time Streaming on Raspberry Pi," Jurnal Infotel, vol. 17, no. 2, pp. 440–455, 2025.

[4] L. Palazzetti et al., "AntPi: A Raspberry Pi Based Edge–Cloud System for Real-Time Ant Species Detection Using YOLO," Ecological Informatics, vol. 91, p. 103383, 2025.

[5] H. Feng et al., "Benchmark Analysis of YOLO Performance on Edge Intelligence Devices," Cryptography, vol. 6, no. 2, p. 16, 2022.

[6] D. K. Alqahtani et al., "Benchmarking Deep Learning Models for Object Detection on Edge Computing Devices," arXiv:2409.16808, 2024.

[7] D. Zhao, "A Comparative Study of Inference Models Based on Image Recognition," Lecture Notes in Electrical Engineering, vol. 1361, 2025.

[8] P. Jajal et al., "Interoperability in Deep Learning: A User Survey and Failure Analysis of ONNX Model Converters," in Proc. ISSTA, 2024.

[9] I. J. Ratul, Y. Zhou, and K. Yang, "Accelerating Deep Learning Inference: A Comparative Analysis of Modern Acceleration Frameworks," Electronics, vol. 14, no. 15, p. 2977, 2025.

[10] "Cross-Platform Optimization of ONNX Models for Mobile and Edge Deployment," White Paper, 2024.

[11] J. L. Mira et al., "Benchmarking of Computer Vision Methods for Energy-Efficient High-Accuracy Olive Fly Detection on Edge Devices," Multimedia Tools Appl., vol. 83, pp. 81785–81809, 2024.

[12] H. Woisetschläger et al., "FLEdge: Benchmarking Federated Learning Applications in Edge Computing Systems," in Proc. Middleware, 2024.

[13] P. D. Rosero-Montalvo, P. Tözün, and W. Hernandez, "Optimized CNN Architectures Benchmarking in Hardware-Constrained Edge Devices in IoT Environments," IEEE Internet Things J., vol. 11, no. 11, pp. 20357–20369, 2024.

[14] Y. Gong et al., "LOTUS: Learning-Based Online Thermal and Latency Variation Management for Two-Stage Detectors on Edge Devices," in Proc. DAC, 2024.

[15] S. H. Choi, J. Kong, and S. W. Chung, "ComBoost: An Instruction Complexity Aware DTM Technique for Edge Devices," in Proc. ISLPED, 2024.

[16] R. Ganesan, "Energy-Aware AI Scheduling for Resource-Constrained Edge Devices," in Proc. ICRCEDA, 2025.

[17] N. Ng et al., "Collaborative Inference in Resource-Constrained Edge Networks: Challenges and Opportunities," in Proc. MILCOM, 2024.

[18] W. A. C. Lopes, M. T. Okano, S. B. Lengua, O. Vendrametto, J. C. L. Fernandes, and M. E. Fernandes, "Evaluating IoT hardware platforms through frugal innovation," South American Development Society Journal, vol. 10, no. 29, pp. 60–80, 2024, doi: 10.24325/issn.2446-5763.v10i29p60-80.

[19] M. T. Okano, H. C. L. dos Santos, E. L. Ursini, M. E. Fernandes, and J. G. C. Gomes, "Open and distance learning: Traditional or frugal innovation?" Contemporary Economics, vol. 17, no. 1, pp. 24–42, 2023, doi: 10.5709/ce.1897-9254.497.

[20] W. A. C. Lopes et al., "Augmented reality applied to identify aromatic herbs using mobile devices," AgriEngineering, vol. 6, pp. 2824–2844, 2024, doi: 10.3390/agriengineering6030164.