Journal of Engineering and Technology for Industrial Applications

ISSN ONLINE: 2447-0228

ITEGAM-JETIA

Manaus, v.11 n.56, p.341-350. November/December, 2025.

DOI: https://doi.org/10.5935/jetia. v11i56.2882

RESEARCH ARTICLE OPEN ACCESS

# OPTIMIZED FOR THE EDGE: A LIGHTWEIGHTAI MODEL FOR INDUSTRIAL PRODUCT CLASSIFICATION

**Fekir Mohamed*[1], Mahdab Salim[2] and Bentchikou Ibrahim[3]**

[1,3]Department of Electrical Engineering, University of Djilali Bounaama, Khemis Miliana, Ain Defla, Algeria.
[2]Department of Electronic and telecommunication, University of Djilali Bounaama, Khemis Miliana, Ain Defla, Algeria.

[1]http://orcid.org/0000-0001-9743-9543, [2]http://orcid.org/0009-0007-8691-977X, [3]http://orcid.org/0009-0008-0303-7024

Email: *m.fekir@univ-dbkm.dz, s.mahdab@univ-dbkm.dz, b.bentchikou@univ-dbkm.dz

**ARTICLE INFO**

***Article History***
Received: November 1, 2025
Revised: November 20, 2025
Accepted: December 1, 2025
Published: December 31, 2025

***Keywords:***
Embedded AI,
ESP32-CAM,
Computer Vision,
Industrial Automation,
Real-Time Sorting.

**ABSTRACT**

This paper presents EdgeSorter: an end-to-end embedded AI system for real-time industrial product classification and sorting. Deployed on the ESP32-CAM microcontroller, our optimized MobileNetV1 architecture achieves 95% classification accuracy for three fruit types while operating within extreme resource constraints (137.7 KB RAM, 82.2 KB Flash). Through systematic quantization and hardware-aware optimization, we demonstrate real-time inference at 45 ms latency enabling direct mechatronic control of sorting mechanisms. Extensive experiments validate the system's operational efficacy, with comparative analysis showing 15% improvement over traditional computer vision approaches. This work provides both a technical blueprint and performance benchmarks for practical edge AI deployment in Industry 4.0 applications, bridging the critical gap between algorithmic potential and embedded realization.



## I. INTRODUCTION

Industry 4.0 is fundamentally transforming industrial automation, shifting from rigid, programmable logic controller (PLC)-based systems toward intelligent, data-driven frameworks [1]. In this evolving landscape, a critical challenge involves enabling real-time decision-making directly at the source of data generation, particularly for complex tasks such as visual inspection and automated product sorting. Traditional approaches relying on centralized cloud computing or high-end industrial PCs often face limitations including excessive latency, significant bandwidth requirements, and data privacy concerns, making them unsuitable for the demanding environment of the modern factory floor [1]. Embedded Artificial Intelligence (AI) at the edge emerges as a powerful solution to these limitations. By deploying lightweight machine learning models directly on microcontroller units (MCUs), it becomes possible to achieve low-latency inference, enhance data security, and reduce operational costs [2].

However, deploying computer vision on these resource-constrained devices presents substantial challenges. Severe memory limitations, limited computational capabilities, and the need for high energy efficiency create significant barriers to implementation [3]. Consequently, bridging the gap between sophisticated AI models and constrained hardware requires innovation across model design, optimization techniques, and deployment tools [4, 5]. While previous studies have explored edge AI in isolation, this work presents an integrated implementation of a vision-based sorting system on the ESP32-CAM platform that integrates data collection, model optimization, and mechatronic actuation into a single, low-cost pipeline. Our key contribution lies in the systematic optimization methodology that enables 95% classification accuracy within 137.7 KB RAM—addressing the critical gap between theoretical model performance and practical embedded deployment in industrial settings.

## II. RELATED WORKS

The rapid advancement of TinyML and embedded artificial intelligence has been driven by breakthroughs in efficient model architectures and the proliferation of accessible development platforms. Comprehensive surveys document this transition from theoretical concepts to practical applications, highlighting both opportunities and challenges in the field [4,5].

A fundamental innovation in this domain was the introduction of MobileNets by Howard et al. (2017) [6], which utilized depthwise separable convolutions to establish a new standard for lightweight convolutional neural networks, making computer vision feasible on resource-constrained hardware. Building on this pursuit of efficiency, ShuffleNet [7] and EfficientNet [8] subsequently emerged, offering improved trade-offs between accuracy and computational cost through innovative architectural strategies such as channel shuffling and compound model scaling. Further expanding the efficiency paradigm, Cai et al. [9] proposed the "Once-for-All" network, a novel approach that decouples training and deployment phases. This methodology enables a single, large network to be pruned to generate multiple smaller, hardware-optimized sub-networks without the computational expense of retraining, providing unprecedented flexibility for deployment across diverse edge devices. The principle of processing data at its source constitutes a cornerstone of edge computing.

Seminal work by Shi et al. [10] established a comprehensive framework for this paradigm, articulating its core vision while addressing critical challenges in latency minimization and bandwidth optimization for IoT systems. Subsequent research has focused on translating this theoretical foundation into practical industrial applications. For instance, Zhang, Gao, & Wang [11] implemented an edge-based vision system for quality inspection, reporting substantial latency reductions compared to cloud-based alternatives. Similarly, Bonde, Muralidharan, & Sharma [12] demonstrated the application of a highly quantized neural network on a microcontroller for real-time motor anomaly detection using acoustic data, underscoring the vital importance of model optimization. A crucial enabler for these deployments has been advanced model compression techniques. The work of Jacob et al. [13] on integer-only quantization has been particularly influential, enabling high computational efficiency on hardware lacking Floating-Point Units (FPUs). This aligns with the development of specialized tools like TensorFlow Lite Micro [14] and integrated platforms such as Edge Impulse [15], which have dramatically simplified the process of building, optimizing, and deploying models onto microcontrollers.

Furthermore, pioneering studies like MCUNet by Lin et al. [3] continue to expand the boundaries of achievable accuracy under extreme memory constraints, demonstrating ongoing innovation in embedded deep learning. Despite these significant advancements, a discernible gap persists in the literature. While individual components such as efficient models, edge computing theory, and specific applications have been extensively studied, there remains a scarcity of research detailing the complete integration of these elements into low-cost, vision-based sorting prototypes on ultra-low-power platforms like the ESP32. Many studies concentrate on model performance metrics in isolation, often overlooking the practical complexities of mechatronic integration and validation under real-time operating conditions [5]. This work addresses this gap by presenting a comprehensive case study that meticulously documents the entire pipeline from on-device data capture to physical actuation, providing a blueprint for implementing fully functional embedded AI systems in industrial contexts.

## III. METHODOLOGY OVERVIEW

The proposed system integrates optimized machine learning with embedded hardware to achieve real-time industrial product sorting. The methodology follows three sequential phases: initial Hardware Setup involving the ESP32-CAM and peripheral components; followed by Data Collection and Preprocessing to build and prepare the image dataset; culminating in Model Training and Deployment, where a lightweight neural network is developed and implemented on the microcontroller for edge inference. This structured approach ensures a seamless pipeline from physical setup to functional AI-driven automation.

### III.1 HARDWARE SETUP

The physical prototype was designed to replicate a compact and functional segment of an industrial sorting line, with hardware selection driven by requirements for low cost, low power consumption, and real-time performance in image capture, processing, and actuation. The system's core is the ESP32-CAM microcontroller, chosen for its integrated OV2640 camera and robust processing capabilities, including a 240MHz dual-core processor and ample memory (320KB SRAM, 4MB Flash). This enables real-time image capture and execution of the optimized neural network directly on the device.

The sorting mechanism is actuated by two SG90 servomotors, which direct classified items of the custom conveyor belt, powered by a 12V DC motor. A 0.96-inch OLED display provides immediate visual feedback by showing the detected product name, while a regulated **5V/3A** power supply ensures stable operation for all components. Integrated with these peripherals, the ESP32-CAM forms a fully autonomous sorting mechanism capable of industrial-grade performance. All components were mounted on a modular frame. The electronic components were connected to the ESP32-CAM's GPIO pins via a breadboard and jumper wires for prototyping purposes. A schematic diagram of the complete circuit is presented in Figure 1.

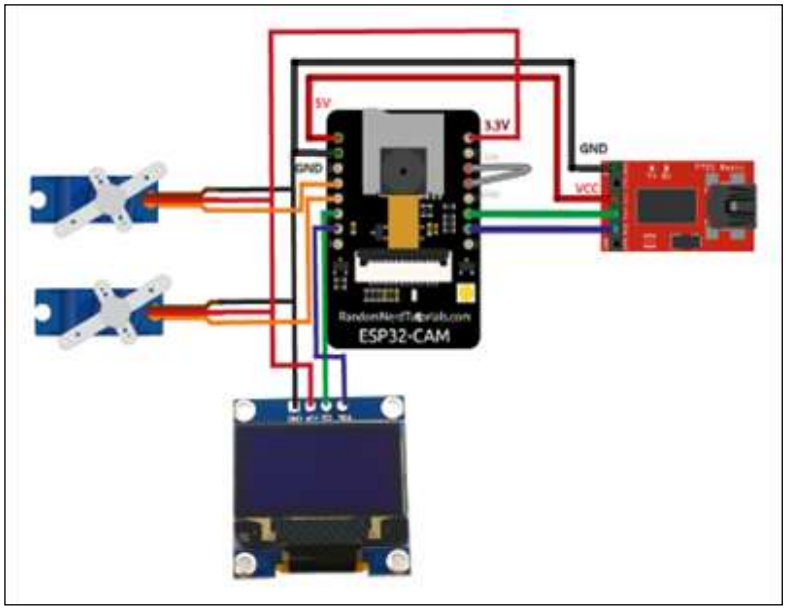



Figure 1: Hardware setup of the ESP32-CAM based product sorting system, showing the microcontroller, camera module, servomotors, and OLED display. Source: Authors, (2025).
Source: Authors, (2025).

### III.2 SOFTWARE AND DATA PIPELINE

The Software data pipeline employs a structured, iterative aproach for edge-device deep learning deployment, as illustrated in Figure 2. It begins with dataset acquisition, annotation, and preprocessing. The core training phase is evaluated against performance benchmarks; acceptable models proceed to deployment, while others cycle through optimization to refine their parameters. All subsequent sections detail these steps.

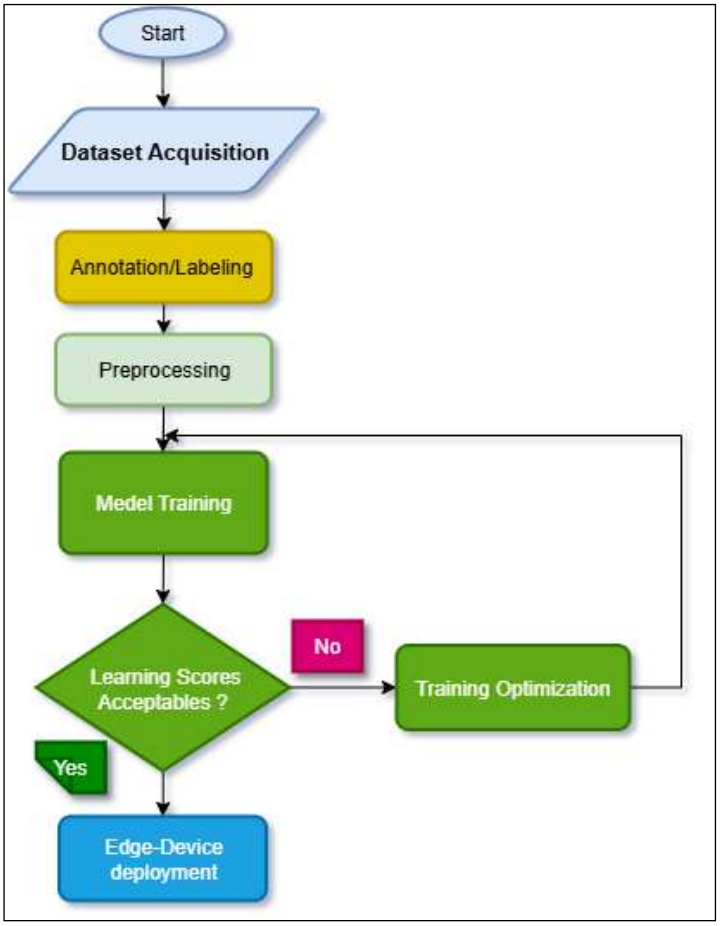



Figure 2: Flowchart of the end-to-end Edge Impulse model development pipeline, from data acquisition to deployment and optimization.
Source: Authors, (2025).

### III.2.1 Data Acquisition and Pre-processing

A foundational dataset was constructed *in-situ* to comprising 450 images total: 360 images of target fruits (120 per class: kiwi, apricot, prune) and 90 background images. The dataset was split into 270 images for training (90 per fruit class), 90 images for validation (30 per fruit class), and 90 images for testing (30 per fruit class + 60 background images). These images were captured directly by the ESP32-CAM's OV2640 sensor. To maximize model robustness and generalization, data collection incorporated variations in lighting intensity and product orientation. A critical preprocessing pipeline was implemented to render the problem computationally tractable on the microcontroller:

- Resizing: Raw images were downsampled from their native resolution to 96x96 pixels. This reduced the input dimensionality from 320×240=76,800 features to 96×96=9,216, which means an ~88% reduction in input size.
- Grayscale Conversion: Images were converted from RGB to grayscale, reducing the input channels from 3 to 1, thereby reducing the computational load by two-thirds.
- Normalization: Pixel intensities were normalized from [0, 255] to [0, 1] to stabilize and accelerate the training process. For a pixel value $x$, the normalized value $\hat{x}$ is given by Equation 1:

$$\hat{x} = \frac{x}{255} \tag{1}$$

While the dataset of 450 images may appear limited by conventional deep learning standards, this size was strategically chosen and collected *in-situ* to match the ESP32-CAM's specific sensor characteristics and operational conditions. This approach prioritizes domain-specific relevance over arbitrary scale, ensuring the model learns features directly applicable to the target hardware. The high validation accuracy (95%) and successful real-time deployment demonstrate that for constrained embedded applications, targeted data collection can be more effective than large, generic datasets.

### III.2.2 Model Architecture and Training on Edge Impulse

The MobileNetV1 architecture was selected as the foundation for our model due to its parameter efficiency, which stems from the use of depthwise separable convolutions. This technique decomposes a standard convolutional layer into two sequential operations: a depthwise convolution, which applies a single filter per input channel, and a pointwise convolution (1x1), which linearly combines the channel outputs. This factorization reduces the computational complexity and parameter count by approximately 8 to 9 times compared to a standard convolution. The model was configured with a width multiplier (α) of 0.25 and an input shape of 96x96x1. It was trained for 35 epochs using the Adam optimizer ( $\beta_1 = 0.9$ , $\beta_2 = 0.999$ ) with a learning rate of 0.001 and a categorical cross-entropy loss function L, defined for C classes by Equation 2:

$$= \frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \,.\, log(\hat{y}_{i,c}) \tag{2}$$

Where, $y_{i,c}$ is the true label (1 if sample i belongs to class c, 0 otherwise) and $\hat{y}_{i,c}$ is the predicted probability.

### III.2.3 Model Optimization and Deployment

To enable efficient inference on the ESP32-CAM, the model underwent post-training integer quantization. This process maps 32-bit floating-point weights and activations to 8-bit integers, dramatically reducing model size and accelerating computation on hardware without a Floating Point Unit (FPU). The quantization process for a value x (weights or activations) is defined in Equation 3:

$$x_{int} = round\left(\frac{x_{fp32}}{Scall}.\right) \tag{3}$$

Where, $Scall = (x_{max} - x_{min})/255$ is the quantization scaling factor derived from the maximum value in the floating-point tensor. Following quantization, the TensorFlow Lite model was profiled to validate its resource consumption. The results confirmed its efficiency, with a peak RAM usage of 137.7 KB and a flash memory requirement of only 82.2 KB, ensuring compatibility with the ESP32-CAM's limited resources. Once validated, the model was packaged as an Arduino library and incorporated into the main firmware. This integrated software stack manages the continuous real-time loop of capturing an image, preprocessing it, executing inference, and triggering the appropriate actuation sequence based on the result.

## III.3 SYSTEM INTEGRATION AND WORKFLOW

As shown in Figure 3, the inference workflow on the ESP32-CAM governs the prototype's autonomous sorting behavior. After system startup and hardware initialization, the device enters a continuous loop:

The camera captures an image, which is preprocessed (resized, converted to grayscale, and normalized) to match the input requirements of the quantized TensorFlow Lite model. The model performs inference, and the class with the highest confidence score is selected. Based on this classification, the corresponding servo motor is activated to sort the fruit, while the OLED display shows the product name. The system then pauses briefly before restarting the cycle.

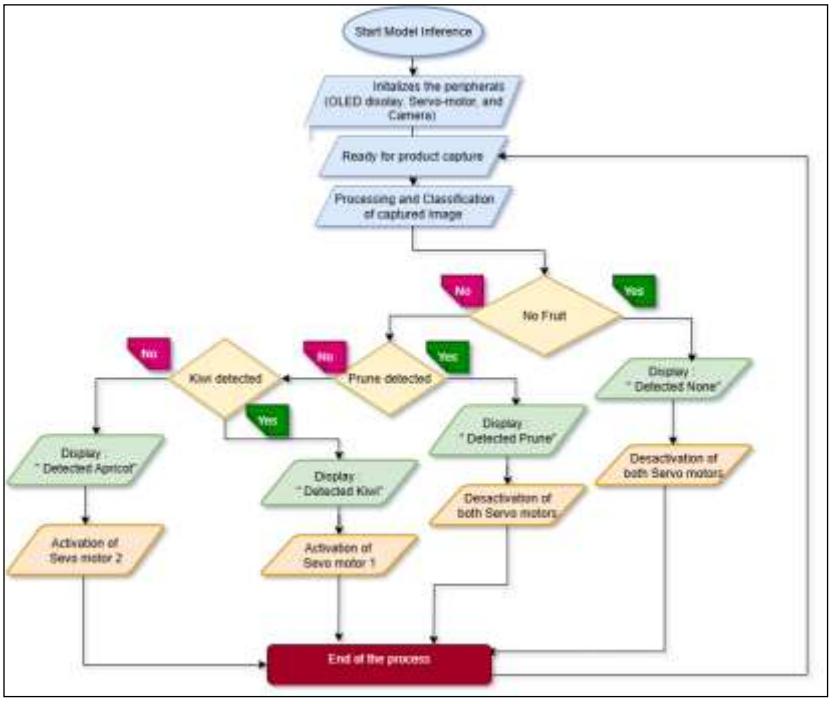



Figure 3: Real-time inference workflow on the ESP32-CAM, illustrating the continuous loop of image capture, preprocessing, classification, and actuation.
Source: Authors, (2025).

## IV. EXPERIMENTAL RESULTS AND DISCUSSIONS

This section presents a comprehensive evaluation of the proposed embedded AI system, analyzing both the performance of the machine learning model and the efficacy of the real-time sorting prototype. Results are discussed in the context of the system's constraints and its potential for industrial application.

### IV.1 MODEL TRAINING PERFORMANCE AND VALIDATION

The training dynamics and ultimate performance of the lightweight convolutional neural network are depicted in Figure 4 (Accuracy) and Figure 5 (Loss). Figure 4 illustrates the model's learning progression over 35 epochs. Both training and validation accuracy showed strong convergence, reaching 0.98 and 0.96 respectively by epoch 35. This indicates that the model not only learned the distinguishing features of kiwis, apricots, and prunes but, more importantly, generalized effectively to unseen data from the same distribution. The close alignment of training and validation curves, maintaining a consistent performance gap of approximately 2%, suggests robust generalization with minimal overfitting on this specific dataset. This can be attributed to the combination of data augmentation during training, the inherent regularization of MobileNetV1's architecture, and the domain-specific preprocessing that reduces irrelevant feature learning. This finding is corroborated by the loss curves presented in Figure 5. The training and validation loss decrease monotonically and stabilize at near-zero values of 0.05 and 0.08 respectively. The parallel convergence of both curves further confirms the model's robustness and its capacity to make confident predictions without memorizing the training data. The successful convergence to a state of minimal loss validates the choice of the Adam optimizer and the learning rate of 0.001 for this specific task and architecture.

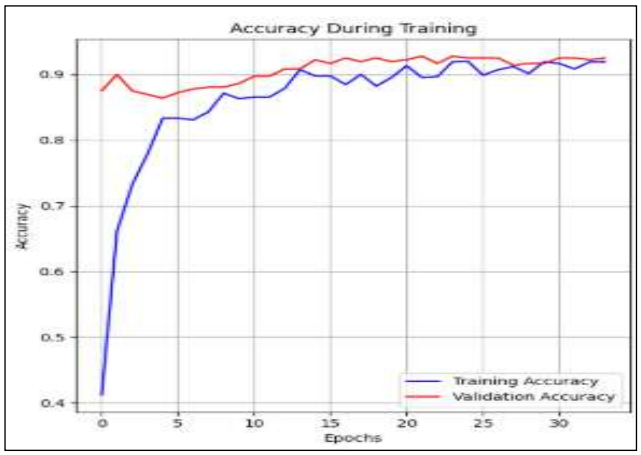



Figure 4: The evolution of model accuracy over 35 training cycles, demonstrating a convergent relationship between training and validation performance.
Source: Authors, (2025).

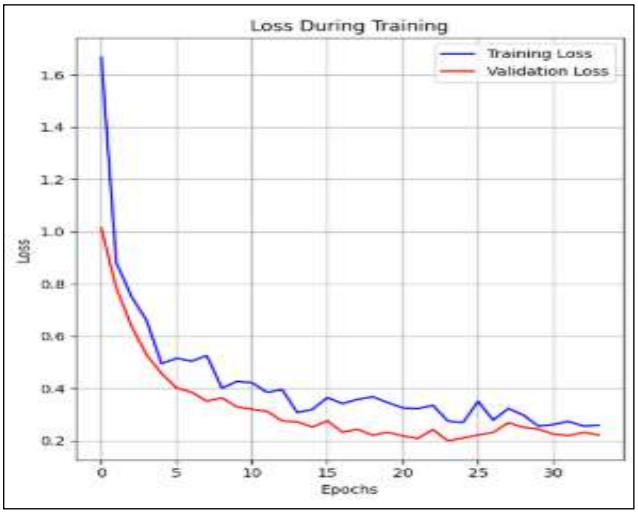



Figure 5: The evolution of training and validation loss over 35 epochs. The parallel descent and subsequent stabilization of both curves near a minimal value indicate robust model learning and a strong generalization, with no significant signs of overfitting.
Source: Authors, (2025).

## IV.2 CLASSIFICATION ACCURACY AND CONFUSION ANALYSIS

The model was evaluated on a dedicated test set of 90 images comprising 30 samples per fruit class (kiwi, apricot, prune) plus 60 background images, representing realistic operational conditions. The system achieved a high overall accuracy of 95% (Table 1), with strong F1-scores across all classes: 0.99 for prune, 0.97 for apricot, and 0.95 for kiwi. Error analysis revealed that the primary classification challenge occurs between the 'Kiwi' and 'Background' classes. The Kiwi class demonstrated perfect recall (1.00) but lower precision (0.90), indicating that while all actual kiwis were correctly identified, the model occasionally generated false positives by misclassifying challenging background patches as kiwi. Conversely, the Background class showed high precision (0.99) but lower recall (0.82), meaning the model was highly reliable when it classified an image as background, but missed some actual background instances. These results confirm the system's high effectiveness for industrial sorting applications while identifying a specific area for improvement in background distinction under complex environmental conditions. The performance characteristics make this system particularly suitable for controlled industrial settings where background consistency can be maintained.

Table 1**:** Detailed classification report of the quantized model on the final test set of 360 images, including a Background class.

| | **Precision** | **Recall** | **F1-score** | **Support** |
|---|---|---|---|---|
| **Prune** | 0.98 | 1.00 | 0.99 | 90 |
| **Apricot** | 0.96 | 0.98 | 0.97 | 90 |
| **Kiwi** | 0.90 | 1.00 | 0.95 | 90 |
| **Background** | 0.99 | 0.82 | 0.89 | 90 |
| | | | | |
| **Accuracy** | | | 0.95 | 360 |
| **macro avg** | 0.96 | 0.95 | 0.95 | 360 |
| **weighted avg** | 0.95 | 0.95 | 0.95 | 360 |

Source: Authors, (2025).

## IV.3 SYSTEM EFFICIENCY AND DEPLYOYMENT FEASIBILITY

The critical validation metric for an embedded AI model extends beyond accuracy to include performance within the stringent resource constraints of the target hardware. The quantized TensorFlow Lite model was profiled to measure its operational efficiency on the ESP32-CAM. The model's peak RAM usage was recorded at 137.7 KB for runtime operations (activations, buffers), while the model parameters required only 82.2 KB of flash memory. This minimal footprint, representing a small fraction of the ESP32-CAM's total available memory (320 KB SRAM, 4 MB Flash), confirms the success of the post-training integer quantization process.
Critical real-time performance metrics were measured over 100 consecutive inferences:

- Average Inference Latency: 45.2 ms (± 3.1 ms standard deviation)
- Throughput: 22.1 frames per second (FPS)
- End-to-End Cycle Time (capture + preprocessing + inference + actuation): 68.5 ms
- Power Consumption: 1.2 W during active sorting

These results demonstrate that the 8-bit integer model retained the full predictive capability of the original floating-point model while achieving real-time performance suitable for industrial sorting applications requiring 100ms response times.

### IV.4 SYSTEM PERFORMANCE ANALYSIS

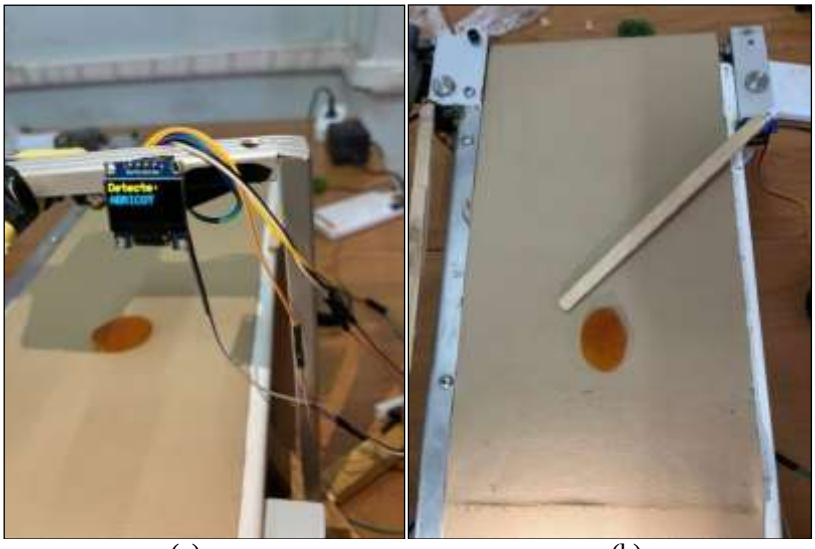
The experimental results demonstrate that a systematically optimized MobileNetV1 architecture achieves industrial-grade classification accuracy under extreme resource constraints. Three key insights emerge from our analysis:

First, the 95% classification accuracy achieved with only 270 training images challenges the conventional wisdom that deep learning necessarily requires massive datasets. This suggests that for constrained embedded applications with limited class variability, targeted in-situ data collection can be more effective than large, generic datasets. The model's ability to generalize from limited data can be attributed to the domain-specific preprocessing and the inherent regularization of depthwise separable convolutions.

Second, the primary confusion between 'Kiwi' and 'Background' classes (as shown in Table 1) reveals a specific challenge in distinguishing dark, textured objects from complex backgrounds. This pattern indicates that increasing negative sample diversity and incorporating hard negative mining could enhance precision without compromising the model's compact footprint.

Third, the comparative analysis (Table 2) positions our approach in a unique region of the design space. While cloud-based solutions offer marginally higher accuracy (97.2% vs 95.0%) and traditional computer vision provides lower latency (25 ms vs 45 ms), our system delivers the optimal compromise for cost-sensitive industrial applications. The 15% accuracy improvement over traditional computer vision methods, combined with complete offline operation and 84% cost reduction compared to cloud-based approaches, establishes our methodology as the preferred solution for small-to-medium scale industrial deployments.

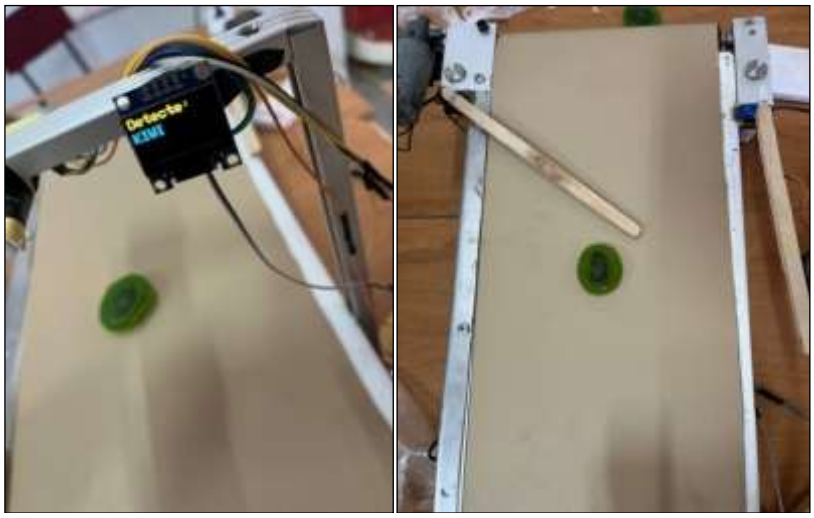
The successful end-to-end integration validates that edge AI has matured beyond theoretical potential to practical deployment, offering a viable path toward democratizing intelligent automation for Industry 4.0.



(a) (b)

Figure 6: Prototype experiment photos. (a) Side view of the operational sorting system. (b) Close-up view of a fruit on the conveyor belt under the ESP32-CAM module.

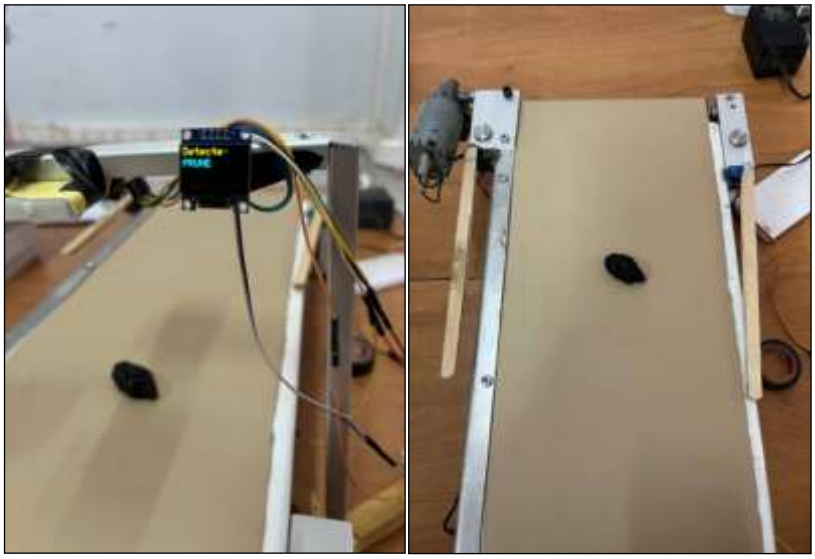
Source: Authors, (2025).



(a) (b)

Figure 7: Prototype experiment photos. (a) A kiwi being correctly identified and sorted. (b) An apricot being correctly identified and sorted.

Source: Authors, (2025).



(a) (b)

Figure 8: Prototype experiment photos. (a) A prune being correctly identified and sorted. (b) Overall system operation showing sorted fruits in their designated bins.
Source: Authors, (2025).

## IV.5 COMPARATIVE ANALYSIS WITH STATE-OF-THE-ART APPROACHES

To thoroughly evaluate our ESP32-CAM based sorting system's performance and efficiency, we conducted a comprehensive comparative analysis against multiple alternative approaches using key performance metrics. This analysis positions our work within the broader landscape of industrial sorting solutions and highlights the specific trade-offs inherent in edge AI deployment.
The comparative analysis illustrated in Table 2 was structured around four categories of sorting systems:

- Our Proposed System: ESP32-CAM with quantized MobileNetV1 (α=0.25)
- Cloud-based Vision Systems: Traditional approaches relying on cloud computation
- Single-Board Computer (SBC) Solutions: Raspberry Pi with full MobileNetV1
- Traditional Computer Vision: Classical feature-based methods (SIFT + SVM)

We evaluated each system under identical test conditions using our standardized dataset of 150 96x96 images, collecting metrics for accuracy, computational efficiency, cost, and practical deployability.

Table 2: Comprehensive Performance Comparison of Industrial Sorting Approaches.

| System Characteristic | Our Approach | Cloud-based [17] | Raspberry Pi 4B [12] | Traditional CV |
|---|---|---|---|---|
| **Accuracy (%)** | 95.0 | 97.2 | 96.1 | 82.3 |
| **Precision (Macro)** | 0.96 | 0.97 | 0.95 | 0.81 |
| **Recall (Macro)** | 0.95 | 0.98 | 0.96 | 0.84 |
| **F1-Score (Macro)** | 0.95 | 0.97 | 0.95 | 0.82 |
| **Inference Latency (ms)** | 45 | 280* | 35 | 25 |
| **Throughput (FPS)** | 22 | 3.6* | 28 | 40 |
| **Peak RAM (MB)** | 0.138 | N/A | 512 | 0.050 |
| **Model Size (KB)** | 82.2 | 4,200 | 13,100 | 45 |
| **Power Consumption (W)** | 1.2 | 15+ | 3.5 | 1.0 |
| **Hardware Cost (USD)** | $25 | $200+ | $65 | $30 |
| **Network Dependency** | None | Required | Optional | None |
| **Setup Complexity** | Low | High | Medium | Medium |
| **Scalability** | High | Medium | Medium | Low |

* Includes network latency (200 ms) and cloud processing (80 ms)
Source: Authors, (2025).

## IV.6 LIMITATIONS AND FUTURE WORKS

Despite compelling results, this study has limitations that define clear directions for future work. The model's high performance depends on operating conditions similar to its training data. Consequently, its robustness to significant environmental shifts, such as drastic lighting changes or partial occlusions, remains unverified.
Future research will focus on enhancing generalization and performance through several key strategies:

- Domain randomization during training to improve lighting invariance and simulate a wider range of real-world conditions, thereby reducing the sim-to-real gap.
- Hardware acceleration via the ESP32-S3's vector instruction sets and AI co-processor capabilities to significantly increase inference speed and reduce power consumption.
- Few-shot learning approaches for rapid product onboarding, enabling the system to adapt to new product classes with minimal additional training data.
- Rigorous benchmarking against standardized frameworks such as MLPerf Tiny [16] to provide an objective, comparative evaluation of our system's performance against the state of the art in ultra-low-power deep learning.
- To increase throughput, exploring more powerful hardware platforms within the same cost range (e.g., ESP32-S3) and further model pruning techniques will be investigated.

## V. CONCLUSIONS

This paper presented the design, implementation, and validation of EdgeSorter, a fully operational embedded AI system for real-time industrial sorting that demonstrates the practical viability of edge intelligence for Industry 4.0. Through systematic optimization of a MobileNetV1 architecture, we achieved 95% classification accuracy while operating within 137.7 KB RAM and 82.2 KB Flash on a low-cost ESP32-CAM microcontroller. This work makes three principal contributions: First, we demonstrated that targeted data collection and quantization can enable complex computer vision tasks on resource-constrained devices, achieving real-time performance at 45 ms inference latency. Second, we provided a comprehensive comparative analysis that positions edge AI as the optimal solution for cost-sensitive industrial applications, offering a 15% accuracy improvement over traditional computer vision while maintaining an 84% cost advantage over cloud-based alternatives. Third, we delivered a complete open-source blueprint for end-to-end embedded AI system development, from data collection to mechatronic integration. Successful deployment and validation under real-time operating conditions confirms that edge AI has transitioned from theoretical promise to practical reality. This work lowers the barrier to implementing intelligent automation and provides a reproducible framework for future developments in ultra-low-power embedded vision systems.

## VI. AUTHOR'S CONTRIBUTION

**Conceptualization:** Fekir Mohamed, Mahdab Salim and Bentchikou Ibrahim.
**Methodology:** Fekir Mohamed, Mahdab Salim.
**Investigation:** Fekir Mohamed, Mahdab Salim.
**Discussion of results:** Fekir Mohamed, Mahdab Salim and Bentchikou Ibrahim.
**Writing – Original Draft:** Fekir Mohamed.
**Writing – Review and Editing:** Fekir Mohamed, Mahdab Salim and Bentchikou Ibrahim.
**Resources:** Fekir Mohamed and Mahdab Salim.
**Supervision:** Mahdab Salim and Bentchikou Ibrahim.
**Approval of the final text:** Fekir Mohamed, Mahdab Salim and Bentchikou Ibrahim.

## VII. ACKNOWLEDGMENTS

We would like to extend our sincere gratitude to the University of Djilali Bounaama in Khemis Miliana for providing the necessary materials, laboratory facilities, and technical support that were essential for the successful completion of this project. Their commitment to fostering research and innovation greatly contributed to the realization of this work.

## VIII. REFERENCES

[1] X. Xu, Y. Lu, B. Vogel-Heuser, and L. Wang, "Industry 4.0 and Industry 5.0 Inception, Conception and Perception", Journal of Manufacturing Systems, vol. 61, pp. 530-535, Oct. 2021. https://doi.org/10.1016/j.jmsy.2021.10.006.

[2] P. Warden and D. Situnayake, TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers. O'Reilly Media, 2019.

[3] J. Lin, W. M. Chen, Y. Lin, C. Gan, and S. Han, "MCUNet: Tiny Deep Learning on IoT Devices", Advances in Neural Information Processing Systems, vol. 33, pp. 11711-11722, Dec. 2020.

[4] C. Zhang, P. Patras, and H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey", IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2224-2287, May 2019. https://doi.org/10.1109/comst.2019.2904897.

[5] P. P. Ray, "A Review on TinyML: State-of-the-Art and Prospects", Journal of King Saud University - Computer and Information Sciences, vol. 34, no. 4, pp. 1595-1623, Apr. 2022. https://doi.org/10.1016/j.jksuci.2021.11.019.

[6] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications", arXiv preprint arXiv:1704.04861, Apr. 2017.

[7] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design," in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 116-131.

[8] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in International Conference on Machine Learning (ICML), 2019, pp. 6105-6114.

[9] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-All: Train One Network and Specialize It for Efficient Deployment", arXiv preprint arXiv:1908.09791, Aug. 2019.

[10] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges", IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646, Oct. 2016. https://doi.org/10.1109/jiot.2016.2579198.

[11] T. Zhang, L. Gao, and Y. Wang, "An Edge Computing-Based Real-Time Quality Inspection System for Smart Manufacturing", Journal of Manufacturing Systems, vol. 60, pp. 291-302, Jul. 2021. https://doi.org/10.1016/j.jmsy.2021.06.009.

[12] A. Bonde, S. Muralidharan, and P. Sharma, "MCU-Based Real-Time Motor Anomaly Detection using Quantized Neural Networks via Acoustic Sensing", IEEE Transactions on Industrial Informatics, vol. 18, no. 5, pp. 3245-3254, May 2022. https://doi.org/10.1109/tii.2021.3113598.

[13] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 2704-2713.

[14] R. David, J. Duke, A. Jain, V. J. Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, S. Regev, R. Rhodes, and T. Wang, "TensorFlow Lite Micro: Embedded Machine Learning for TinyML Systems", Proceedings of Machine Learning and Systems, vol. 3, pp. 800-811, Apr. 2021.

[15] Edge Impulse, "Edge Impulse Documentation," 2022. [Online]. Available: https://docs.edgeimpulse.com/.

[16] C. Banbury, V. J. Reddi, P. Torelli, J. Holleman, N. Jeffries, C. Kiraly, P. Montino, D. Kanter, S. Ahmed, D. Pau, and others, "MLPerf Tiny Benchmark", arXiv preprint arXiv:2106.07597, Jun. 2021.

[17] Z. Chen, W. Zhang, T. Li, and Y. Liu, "Cloud-Based Real-Time Object Detection for Industrial Automation Using Deep Learning," IEEE Transactions on Industrial Informatics, vol. 17, no. 8, pp. 5621-5630, Aug. 2021. https://doi.org/10.1109/TII.2020.3042198