Journal of Electrical Engineering and Computer (JEECOM)
Vol.8, No.2 (2026), DOI: 10.33650/jeecom.v8i2.17208
p-ISSN: 2715-0410; e-ISSN: 2715-6427

Real-Time Packaging Quality Inspection Using YOLOv8 with ESP32-Based IoT Actuation and Monitoring

Muhammad Arya Saputra[1], As'ad Shidqy Aziz[2]*
[1,2] Department of Electrical Engineering, Universitas Negeri Surabaya, Indonesia

Article Info

Article history:

Received September 2, 2026
Revised September 21, 2026
Accepted October 18, 2026

Keywords:

YOLOv8;
ESP32;
Internet of Things;
Product Inspection;
Real-Time System

ABSTRACT

Manual product inspection and sorting in small and medium-sized enterprises (SMEs) are often slow and error-prone, whereas conventional computer vision systems can require costly infrastructure. This study develops a low-cost, real-time packaging inspection and sorting prototype in which YOLOv8 inference is executed on a laptop, while an ESP32 handles position sensing, servo actuation, and IoT communication. The closed-loop system integrates visual detection, physical sorting, local LCD monitoring, Blynk, Google Sheets, and end-to-end latency characterization. The dataset comprises 1,510 packaging images divided into Good Product and Damaged Product classes. Five train-validation-test split scenarios were evaluated using early stopping, with model performance assessed using precision, recall, mAP@0.5, and mAP@0.5:0.95. The best configuration achieved an mAP@0.5 of 97.6%; at the selected evaluation threshold, no false negatives and two false positives were observed across 292 evaluated instances. The main sorting response time was approximately 0.51 s, providing a 1.40 s real-time margin at a conveyor speed of 30 RPM. Physical sorting success rates were 86% for Damaged Products and 96% for Good Products, while 50 of 50 logging trials were transmitted successfully under the evaluated network conditions. These results indicate the feasibility of the proposed YOLOv8-laptop-ESP32 architecture for real-time automated inspection and sorting under the evaluated laboratory conditions, while also showing that mechanical and synchronization factors remain important for further system improvement.

Corresponding Author:
As'ad Shidqy Aziz,
Universitas Negeri Surabaya, Ketintang Wiyata Street, Surabaya, 60231, Indonesia
E-mail: asadaziz@unesa.ac.id; Orcid ID: https://orcid.org/0009-0008-5705-0363

1. INTRODUCTION

Advancements in AI vision and machine learning technologies have significantly impacted industrial transformation, particularly by driving the automation of production, inspection, and quality control processes across manufacturing and agro-industrial sectors [1], [2]. Nevertheless, packaging quality inspection and sorting remain manual or semi-automated in many production settings, highlighting the need for automated systems that can detect visible packaging defects and separate Bad Products from acceptable ones. This situation gives rise to several issues, including reliance on human labor, high error rates caused by fatigue or evaluator subjectivity, and inconsistent or slow inspection results especially in complex environments characterized by uneven lighting, small and densely packed objects, occlusion, or blurred images [3], [4]. A systematic review of AI-ML applications in the food processing industry indicates that conventional manual inspection methods achieve an accuracy of only around 80–90%, whereas AI vision and machine learning approaches can boost accuracy to over 99%, while simultaneously reducing machine downtime from approximately 10% to just 2% and cutting labor costs by up to 30% [2].

Large-scale industrial automation solutions based on computer vision have been widely implemented and proven effective such as real-time small object detection frameworks for monitoring compliance in industrial environments, achieving mAP exceeding 0.93. However, infrastructure complexity including the need for high-end GPUs and large-parameter models like RT-DETR-L (exceeding 60 MB with dozens-of-times higher computational load) along with high implementation costs, limits adoption among resource-limited SMEs [5].Advanced versions such as YOLOv10, YOLOv11, and transformer-based models like RT-DETR have been reported to offer improved accuracy in their respective studies, but typically incorporate additional architectural modules or complex attention mechanisms that increase training time and computational demands [10].

YOLOv8, with its anchor-free design and decoupled head for objectness, classification, and regression, remains a widely adopted baseline easily adaptable into lightweight variants for resource-constrained deployment, such as YOLOv8n-ShuffleNetv2-Ghost-SE (2.6 MB) [9] and SE-ATT-YOLO (50 FPS) [11]. Given these characteristics, YOLOv8 was selected as a practical option for low-cost embedded systems like the ESP32 and standard webcams.

Previous research has advanced YOLOv8-based detection across contexts.[6] developed SDG-YOLOv8 for highway domain shift, untested in industrial defect detection [7] designed GFL-Net for small object detection in drone imagery, untested in real-time conveyor inspection with actuator integration; another study proposed YOLO-D for low-light robustness, limited to dark-to-light shifts and unvalidated on physical sorting hardware [8] developed Dynamic-YOLO for metal surface defect detection with cross-hardware latency benchmarking, but limited to lab-scale datasets without physical actuator or IoT integration. A systematic review [9] of 48 YOLO studies found only 8 (16.7%) validated under representative commercial-scale conditions. To the best of our knowledge, no existing study has combined a YOLOv8-based packaging defect detection system robust to real-world domain gaps with physical sorting actuators and IoT monitoring in a single, operationally validated workflow.

To address this gap, this study aims to design, implement, and operationally validate an integrated YOLOv8-based packaging defect detection and sorting system combining real-time detection, physical actuation, and IoT-based monitoring in a single closed-loop workflow suitable for low-cost embedded deployment. This article makes three key contributions: first, an integrated closed-loop architecture linking YOLOv8 detection, ESP32-servo actuation, and dual logging (LCD and IoT platforms Blynk/Google Sheets), establishing an operationally validated workflow rather than a standalone detection model; second, characterization of end-to-end latency from image acquisition to sorting gate actuation, quantitatively validating real-time performance rarely measured comprehensively in similar studies [10]. and third, evaluation of train-validation-test split ratios' impact on detection accuracy, following the approach in [11].

2. METHOD

This research comprised several stages: system design, image acquisition and annotation, YOLOv8 model training, performance evaluation, and integration with hardware and an IoT-based monitoring system. The workflow follows a standard deep learning development cycle involving requirements identification, data collection, preprocessing, training, testing, and hardware implementation [12].

2.1. System Overview and Hardware Setup

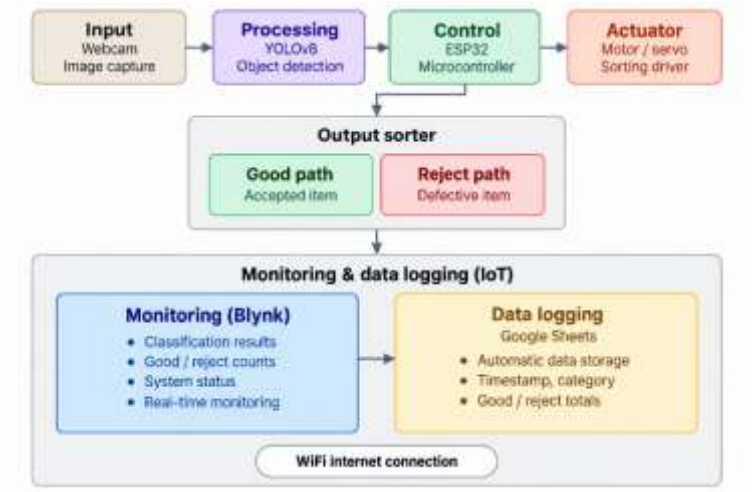

Figure 1. System Block Diagram

The proposed system adopts a closed-loop architecture integrating two synchronous subsystems: a YOLOv8-based computer vision module and an ESP32-based actuation and monitoring module [13], [14].

ISSN: 2715-6427

The classification results are transmitted to the ESP32 via serial or Wi-Fi to control a servo that directs products to the appropriate lane. Detection and actuation are separated into two layers to decouple decision-making from physical execution. Core processes operate locally to maintain sorting during network disruptions, while LCD, Blynk, and Google Sheets provide supplementary monitoring and data logging [15],[16].A non-blocking control strategy enables sensing, classification, actuation, display updates, and IoT transmission to operate concurrently, maintaining system responsiveness and continuous conveyor operation.

Communication with the ESP32 is implemented via an asynchronous HTTP GET request sent on a separate thread so it does not block the detection loop. To improve robustness against transient network issues, the request is retried up to 2 times with a 3-second timeout per attempt before being logged as a failed transmission; sorting decisions themselves are not blocked by a failed transmission, since actuation logic runs independently of the logging/monitoring path.

Table 1. Main Components

System Components	Main Functions	Roles in the System
Product conveyors and transport mechanisms	Transport products through detection zones consistently	Maintain relatively uniform product positioning and orientation during the image acquisition and sorting process
Image acquisition module	Taking product images for analysis	The camera is positioned at a fixed position and height to produce consistent imagery for the detection model
Object presence/position detection sensor	Detecting the presence of a product at a specific point	Determine the time of image capture and ensure the actuator works according to the position of the product; Pull-up and debounce configurations are used to reduce dual triggering
Processing and handling unit (ESP32)	Processing sensor signals and controlling actuators and communications	Execute system logic, set up sorting processes, and provide wireless connectivity for monitoring
Actuator Output (Servo Motor)	Actuating the product director mechanism	Diverting products to a line or container based on the results of the classification; Nonblocking controls support repetitive operation without interfering with other processes
Local display interface (LCD)	Displays system operational information	Displays categories and the number of products that have been sorted in real-time
IoT connectivity and monitoring module	Send and store sorted data	Provides remote monitoring over Wi-Fi networks as well as data synchronization with cloud platforms
YOLOv8 detection model	Detect and classify products	Perform real-time object detection with computational accuracy and efficiency in mind

Table 1 summarizes the main components used in a self-propelled sorting system Computer Vision and IoT and their respective functions. The system integrates conveyors, cameras, sensors, ESP32, servo motors, LCDs, IoT connectivity, and YOLOv8 in a single process flow to support automatic product detection, classification, sorting, and monitoring. The integration of such components allows the detection and sorting process to be carried out in real-time with wireless communication support for system monitoring [14], [15], [16], [17].

Table 2. Hardware and Software Specifications

Component	Specification
Camera	JTW9 USB webcam — FHD 1920×1080 (1080p/30fps or 720p/30fps, fixed focus), 1/6.9" HD color CMOS sensor, aperture f/2.6, SNR ≥60dB, USB 2.0 interface (150cm cable), input 5V/500mA, 112g
Inference computer	Windows laptop/PC, CPU-only inference (device='cpu', 4 threads); AMD Ryzen 7 5825U with Radeon Graphics (2.00 GHz base clock), 8.00 GB RAM (5.85 GB usable); external USB webcam (index 1, DirectShow) at 640×480 @ 30 FPS
Microcontroller	ESP32-WROOM-32 development board (38-pin generic ESP32 DevKit), dual-core Xtensa LX6, built-in Wi-Fi/Bluetooth
Infrared sensor	IR obstacle avoidance sensor module (MH-8 type), IR emitter/receiver pair with onboard comparator and adjustable sensitivity potentiometer, 3-pin OUT/GND/VCC digital output

Component	Specification
Servo motor	Tower Pro MG996R servo motor, metal-gear high-torque, operating voltage 5V (Red = +5V, Brown = GND, Orange = PWM signal)
Training platform	Google Colab (GPU runtime, CUDA device 0), NVIDIA Tesla T4 (15,360 MiB VRAM), CUDA 13.0, driver 580.82.07
Training framework	Ultralytics YOLOv8 (pip install ultralytics), version 8.4.47; Batch size = 16; Image size = 640×640

2.2. Dataset Collection and Annotation

The dataset used consisted of 1,500 images of empty product packaging collected in a laboratory environment. The packaging had no logo or brand identity, so the detection process focused on the physical condition of the packaging. The dataset consisted of two classes, namely Good Products and Bad Products. The Bad Product class comprised 785 images (52.33%), while the Good Product class comprised 715 images (47.67%). Image acquisition was conducted with variations in lighting, object orientation, camera angle, and packaging Bad Products characteristics to increase data diversity. Variations in imaging conditions and defect characteristics were also considered in the development of datasets for defect-based detection in Computer Vision [18], [19]. Each image was annotated using bounding boxes on the object area and grouped into two predefined classes. Bounding box annotation was used to determine the location of objects in the image and was a common approach applied in object detection tasks [20]. The annotated dataset was then divided into Training, Validation, and Testing data according to the partition scenario used in the study.

2.3. Training model (including split scenarios)

The YOLOv8 model was trained for object detection and classification using the annotated dataset, divided into Training, Validation, and Testing subsets. This three-way split follows prior YOLOv8-based research, such as [19] (80:10:10) and Chaman et al. (2026) (70:20:10). This study used five dataset split scenarios—70:20:10, 80:10:10, 75:15:10, 60:20:20, and 65:20:15—with all scenarios using the same dataset and configuration, varying only the split proportion as the comparative variable, consistent with [20], with a consistently retained subset of data across each experiment to allow for performance comparison.

Training was capped at 100 epochs, following the standardized configuration used in [21], for comparing YOLO-based models [19] also applied an SGD optimizer with fine-tuning on a pre-built, annotated dataset. To ensure consistent evaluation across scenarios, model performance was assessed using Precision, Recall, mAP50, and mAP50-95—metrics similarly applied in Chaman et al. (2026) for standardized YOLO model evaluation.

2.4. YOLOv8

YOLOv8 is a one-stage detection model designed for fast inference without sacrificing accuracy, structured around a backbone, neck, and detection head. The backbone's C2f module strengthens feature flow with minimal added computational cost [31][22]. while the PAN-FPN neck preserves semantic and spatial information across scales [23]. This study uses the YOLOv8n variant, chosen for its balance of accuracy and inference speed suited to a low-cost, real-time sorting system.

The model was trained using the Ultralytics YOLOv8 framework (v8.4.47), initialized from COCO-pretrained weights (yolov8n.pt), with a batch size of 16, input resolution of 640×640 pixels, and a maximum of 100 epochs with early stopping (patience = 20). The optimizer and learning rate were left at Ultralytics' automatic default ("auto"); the resolved values were not retained after training and are noted here as a reproducibility limitation. Training was performed on a Google Colab GPU runtime (NVIDIA Tesla T4, 15,360 MiB VRAM, ~12 GB system RAM, Linux-based environment).

The detection head is decoupled, separating bounding box regression from classification, and anchor-free—directly predicting center points, object dimensions, and class probabilities without predefined anchor boxes, improving adaptability and computational efficiency [24]. Available in multiple variants of differing depth and width, YOLOv8 allows flexible trade-offs between accuracy, speed, and resource demands, establishing it as a strong foundation for real-time object detection systems [21].

2.5. System Integration and IoT Deployment

The system operates continuously: as packaging passes through the camera's coverage area, YOLOv8 detects and classifies it (Good/Bad Products) with bounding box output, consistent with [25] s use of visual-IoT integration for abnormal event detection. An infrared sensor then validates the packaging's position, timing the sorting mechanism's activation in line with.[26] Based on the validated classification, the system diverts Bad Products via the actuator while Good Products proceed unimpeded [27]. and both statuses are logged through the IoT gateway for remote monitoring and trend analysis, consistent with industrial IoT practice [28].

ISSN: 2715-6427

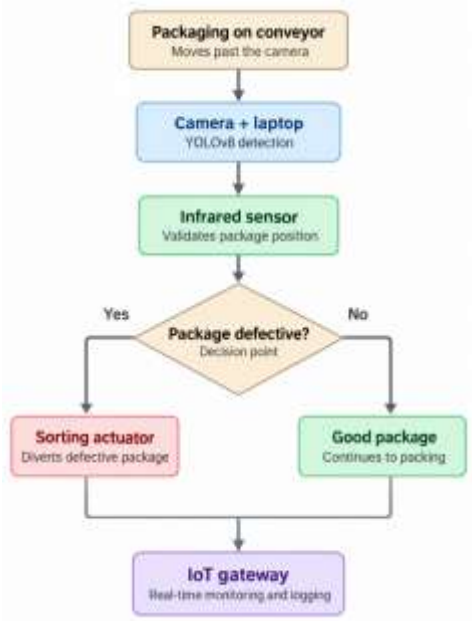

Figure 2. Flowchart System Integration and IoT Deployment

To avoid single-frame errors, the system applies per-class confidence thresholds (0.45–0.75 for Bad Products, 0.55 for Good Product) and evaluates a rolling 20-frame history, locking a decision only when Bad Products frames reach ≥60% of the last 20 (min. 8) or when a 15-sample majority vote reaches ≥12. Once locked, the result is sent to the ESP32 via HTTP GET (2 retries, 3-second timeout), with the state held for 45 frames to prevent flickering before a new object is accepted.

3. RESULTS AND DISCUSSION

This section presents the training and testing results of the YOLOv8 model across five dataset split scenarios, along with integration test results for the ESP32-based sorting system and IoT platform, supported by tables, graphs, and comparison with prior research. It covers: an overview of training results, comparison of the five split scenarios, training dynamics and early stopping, confusion matrix and error analysis, hardware/IoT integration results, and comparison with related studies.

3.1. General Description of Training Results

The final dataset comprised 1,500 product packaging images, preprocessed before training. Auto-Orient corrected image orientation based on EXIF metadata for consistency, while resize standardized inputs to 640×640 pixels via the stretch method, per YOLOv8's architectural requirements. To improve data variation and generalization, augmentation produced three outputs per training image, using horizontal/vertical flips, 90° rotations (clockwise and counterclockwise), and random rotation (-15° to +15°)simple affine geometric transformations proven to consistently improve generalization across computer vision tasks while remaining computationally simple compared to more advanced augmentation methods [29]. Augmentation was applied only to the training subset to prevent data leakage, ensuring validation and testing were performed on entirely unseen data.The preprocessed and augmented dataset was then split into five training/validation/testing scenarios with varying proportions, as described in the Methods chapter. Training was capped at 100 epochs with early stopping (patience = 20), halting automatically if validation performance showed no improvement for 20 consecutive epochs. A summary of the termination epochs for each scenario is shown in the table below.

Table 3. Datasheet ratio division

Scenario	Ratio	The epochs stops	Remarks
V1	80:10:10	52	Early stopping is active
V2	70:20:10	100	Reaching the maximum limit of the epochs
V3	60:20:20	84	Early stopping is active
V4	75:15:10	85	Early stopping is active
V5	65:20:15	100	Reaching the maximum limit of the epochs

As shown in Table 3, scenarios V2 (70:20:10) and V5 (65:20:15) ran the full 100 epochs without triggering early stopping, indicating the model had not yet fully converged, whereas V1, V3, and V4 stopped earlier (epoch 52, 84, and 85) once validation performance plateaued for 20 consecutive epochs confirming the mechanism worked as intended to prevent overfitting while reducing training time.

3.2. Performance Comparison of 5 Dataset Split Scenarios

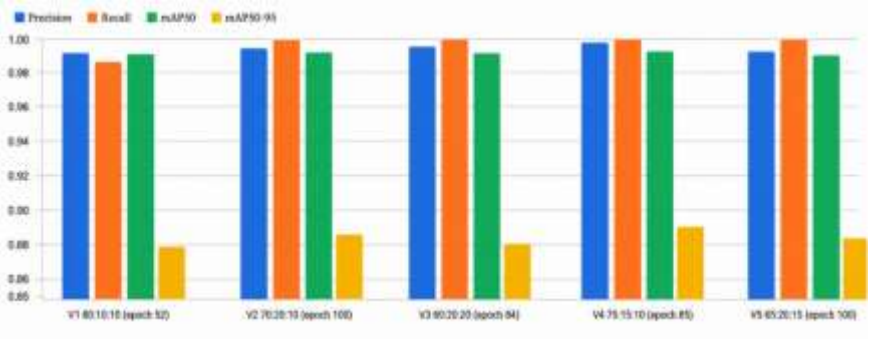

Figure 3. Performance Comparison 5 scenarios

These findings align with prior research highlighting dataset split ratio as a key factor in deep learning-based detection.[30] used a 70:20:10 split to balance training and validation, similar to this study's V1, though the present results show it is not the only configuration capable of yielding optimal performance.[31] used 80:10:10, assuming more training data improves YOLOv5 performance, but this study's results challenge that: V2 (80:10:10) achieved high recall (1.000) but lower mAP50-95 than V4, suggesting a larger training proportion does not automatically ensure better generalization.

For dynamic road-object detection [32] applied a 70:10:20 split and achieved strong performance (99.5% accuracy, 97.6% mAP@0.5), with a larger testing-than-validation proportion—consistent with this study's V4 scenario, supporting more representative evaluation. adds that performance depends more on dataset distribution and characteristics than training volume, while.[34], found that split strategies ignoring inter-instance independence (data leakage) can inflate performance estimates.These findings support the interpretation that V4's advantage stems from a larger validation/testing proportion rather than a larger training share, consistent with the dataset's relatively low complexity (two classes: Good and Defect). In conclusion, no single split ratio is universally optimal—the ideal ratio depends on dataset characteristics, class count, and object complexity, as shown through comparison with [30], [31], [32], [33],[34].

3.3. Training Dynamics and the Early Stopping Effect

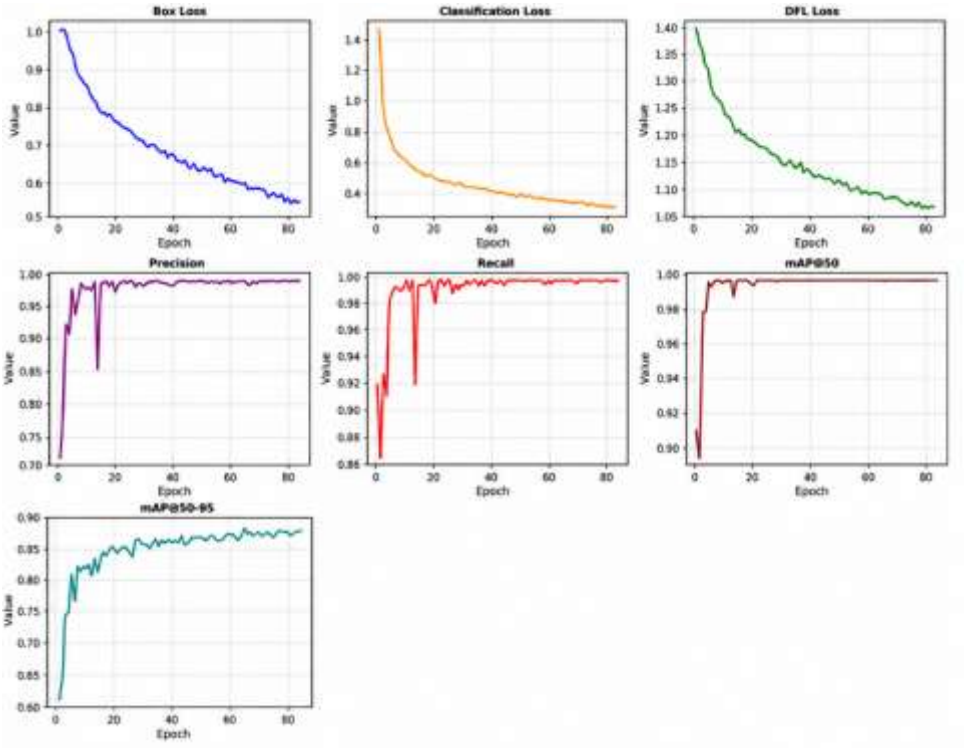

Figure 4. Training Process Graphics

ISSN: 2715-6427

The YOLOv8 training graph in Figure 4. shows a healthy convergence pattern: box loss, classification loss, and DFL loss decreased sharply in the first 20 epochs and then sloped; Precision and recall have stabilized above 0.98 since the 20th epochs; mAP@50 reach the plateau in the epochs range 15–20; While mAP@50-95 the most stringent metric because it demands high localization precision only really sloped in the 60–70 epochs range before training was automatically stopped at the 85th epochs (from the maximum limit of 100 epochs).

3.3.1 Theoretical Justification for Early Stopping

This pattern aligns with the theoretical basis of early stopping as a form of implicit regularization, where training halts once the patience-based generalization-loss criterion is triggered [35] Similar behavior has been reported in comparable YOLOv8 studies, where training capped at up to 1,000 iterations was stopped early after 20 epochs without improvement, without compromising final detection performance (precision 0.90–0.93, recall 0.95–0.97) [36], consistent with the training efficiency achieved here (~1.335 hours versus a projected ~1.57 hours for the full 100-epoch run).

3.3.2 Emperist Evidence from Similar YOLOv8 Studies

These findings are consistent with [36]where YOLOv8 and Mask R-CNN were configured for up to 1,000 iterations but stopped early if validation performance showed no improvement for 20 consecutive epochs minimizing overfitting while improving generalization. The optimal epochs count was determined via preliminary tests, selecting the point where validation accuracy plateaued or declined, matching the pattern observed here in mAP@50–95 around the 60–70 epochs range.The same study showed early stopping (patience = 20) was practically effective: YOLOv8 maintained strong final performance (precision 0.90–0.93, recall 0.95–0.97 depending on dataset) without running the full 1,000 iterations, while achieving high inference speed (128 FPS for single-class segmentation) relevant to real-time applications confirming that early termination does not compromise model quality, consistent with the training efficiency of 1.335 hours achieved here versus a projected 1.57 hours for a full 100 epochs run.

3.4. Confusion Matrix and Error Detection Analysis

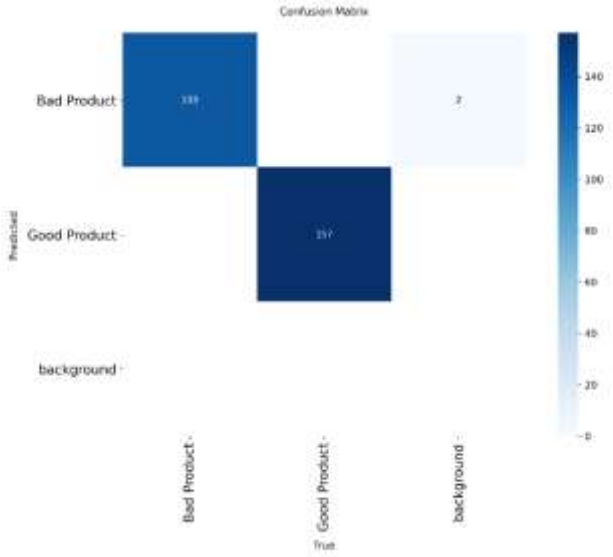

Figure 5. Confusion matrix

Figure 5 and Table 4 show the detection confusion matrix. Rows represent predictions, columns represent ground truth, and Background entries represent unmatched predictions or objects. The matrix contains 290 annotated objects: 133 Bad Products and 157 Good Products. Two unmatched Bad Product predictions bring the prediction count to 292; this is not the number of test images.

Table 4. Detection confusion-matrix counts

Predicted/True	Bad Products	Good Product	Background
Bad Products	133	0	2
Good Product	0	157	0
Background	0	0	0

All 290 annotated objects were matched to the correct class. The Background column contains two false positives, while the Background row indicates no false negatives at the evaluated settings. Background regions are not assigned arbitrary true-negative counts, so an accuracy metric derived from such counts is omitted

Table 5. Confusion matrix results

Class	TP	FP	FN	Precision	Recall	F1-score
Bad Products	133	2	0	98.52%	100%	99.25%
Good Product	157	0	0	100%	100%	100%

Precision measures correct predictions, recall measures coverage of annotated objects, and F1-score is their harmonic mean (Equations 1–3). Ratios are multiplied by 100 for percentages. The displayed matrix is reported at confidence = 0.30 and matching IoU = 0.45; the latter is distinct from an IoU threshold used for non-maximum suppression.

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (3)$$

The derived Bad Products precision is 133/135 = 98.52%, recall is 100%, and F1-score is 99.25%. Good Product precision, recall, and F1-score are 100%. Across both classes, micro-averaged precision is 290/292 = 99.32%; this is reported as precision rather than classification accuracy. mAP@0.5 averages class-specific average precision at IoU = 0.5, whereas mAP@0.5:0.95 averages over IoU thresholds from 0.5 to 0.95 in steps of 0.05. Unlike a confusion matrix at one operating point, average precision summarizes the precision–recall curve. Matching in a detection confusion matrix also depends on localization through IoU.

3.4.1. Failure case analysis

Among the 290 annotated objects, no false negatives were observed, indicating that all Good and Bad Products were successfully detected at the selected operating threshold. However, two additional unmatched detections were recorded as false positives, in which background regions were incorrectly classified as Bad Products. These errors may be associated with visual interference from conveyor surface textures, reflections, and shadows that resemble defect-related features, particularly under non-uniform illumination. Although no false negatives were observed in the present test set, the single-view camera configuration remains a potential limitation because defects located on non-visible surfaces may not be captured. This issue should therefore be considered in future

Small-scale defects pose a further risk, as their weak feature representation on the feature map increases the likelihood of false negatives under suboptimal lighting or resolution, consistent with findings that very small objects struggle to propagate effectively through early convolutional layers. Overall, the very low error rate (2 of 292 instances, ≈0.68%) reflects strong generalization on the test data, though the residual false positives warrant visual inspection to ensure the system doesn't mistakenly halt production over false Bad Products detection on clean surfaces.

3.5. System Integration Results

After the performance of the YOLOv8 model is evaluated separately, this section discusses the end to end performance of the system, starting from the product image captured by the webcam to the product being physically sorted and the data recorded on the IoT platform. This test is important to answer whether a system that individually has high detection performance is also capable of performing in real-time when all components cameras, YOLOv8 models, ESP32, servo, Blynk, and Google Sheets are integrated into a single whole.

3.5.1. Testing Each System Stage

Testing for each stage is done by measuring the time required at each stage of processing, from model inference to data updates on cloud services. The measurement results are shown in Table 5.

Table 6. System testing

Yes	Stages	Time (seconds)	Remarks
1	YOLOv8 Inferences	0,16	Detection & classification of objects per frame
2	Data delivery to ESP32	0,05	Classified serial/WiFi communication
3	Servo actuation	0,30	Servo movement directing product
4	Blynk display update	1,08	Real-time monitoring via smartphone
5	Google Sheets Storage	1,30	Data logging via Google Apps Script

ISSN: 2715-6427

Based on the Table, the total accumulated time of all stages is 2.89 seconds. However, that time does not represent the overall sorting latency because the Blynk monitoring process (1.08 seconds) and Google Sheets storage (1.30 seconds) run after the sort decision and does not hinder the actuation. The system's main response time, which includes YOLOv8 inference (0.16 seconds), data transmission to ESP32 (0.05 seconds), and servo actuation (0.30 seconds), is approximately 0.51 seconds. These results show that the system is able to quickly detect and de sort physical processes, while the monitoring and data logging processes take place separately.

3.5.2. Physical Sorting Success Testing

Physical sorting testing is performed to determine system level accuracy, which is the overall success rate of the system in directing the product to the right path in contrast to the accuracy of the YOLOv8 model which only measures the accuracy of image classification. The test was carried out 100 times with the results as shown in Table 6.

Table 7. Sort success testing

Product Type	Number of Tests	Successful	Fail	Percentage
Bad Products	50	43	7	86 %
Good Product	50	48	2	96 %

The test results showed 9 sorting failures across Bad and Good Product categories—a success rate notably lower than the YOLOv8 model's near-100% evaluation accuracy, confirming that system-level performance is shaped by factors beyond the model itself, namely mechanical and real-time constraints. Likely contributors include delays in transmitting detection data to the ESP32, causing the servo to lag behind the product's movement; positional variation on the conveyor misaligning servo thrust; and network delays when monitoring data transmission overlaps with the sorting process.

To quantify statistical uncertainty, 95% confidence intervals were calculated using the Wilson score method: Bad Products 86% (43/50), CI 73.8%–93.0%; Good Product 96% (48/50), CI 86.5%–98.9% (verified via statsmodels). The relatively wide interval for Bad Products, given n = 50 per category, suggests a larger sample size is recommended in future testing. To identify the dominant cause of the model-to-system performance gap, each of the 9 failures was reviewed and classified by dominant cause, as summarized in Table 8.

Table 8. Classification of Sorting Failure Causes

Cause	Cases	% of Failures
ESP32 communication delay	0	0 %
Conveyor speed exceeding servo response	0	0 %
Misaligned product position	3	33,3%
Network delay from concurrent monitoring transmission	6	66,7 %
Total	**9**	**100%**

The classification results show that the dominant cause of failure was network delay from concurrent monitoring transmission, accounting for 66.7% of the 9 recorded failures (with the remaining 33.3% attributable to misaligned product position), confirming that the model to system performance gap is primarily attributable to communication-related factors rather than limitations of the YOLOv8 detection model itself.

3.5.3. Dual Logging Results (Local LCD vs Blynk/Google Sheet)

After the system is designed with a dual logging mechanism, namely the display of data locally through a 16×2 I2C LCD installed directly on the ESP32, as well as remote recording through the Blynk and Google Sheets applications. The purpose of this design is to ensure that the system can still display information on the number of Good and Bad Products locally even if the internet connection is lost, as described in the methods chapter.

Table 9. Dual logging test results

Parameters	Value
Number of Tests	50
Data successfully sent	50
Data failed to be sent	0
Success rate	100 %

In addition, the average data update delay in the Blynk application was recorded at 1.08 seconds, while the data storage delay in Google Sheets was recorded at 1.30 seconds. These two values indicate that under normal network conditions, the system is able to perform stable remote logging with relatively little delay and does not interfere with the sorting process that runs in real-time.

3.6. Comparison with Previous Research

This study develops an integrated YOLOv8-based inspection and sorting system for Good and Bad Products, combining an ESP32-controlled servo, infrared sensors, and LCD, Blynk, and Google Sheets for monitoring—addressing a gap in prior YOLO research that focused on detection alone without physical sorting integration. The model achieved 97.6% mAP@0.5, with precision and recall above 98% across 292 instances (no false negatives, two false positives). However, physical sorting success reached only 86% for Bad Products and 96% for Good products, showing that model accuracy does not always translate into physical performance due to factors like communication delay, conveyor speed, and servo response.

Table 10. Comparison of previous research

Research	Research Object	Methods/Models	Key Results	Focus/Advantages
Zhu et al. (2023)	Crown wheel surface defect detection	YOLOv8 + preprocessing + augmentation	mAP reaches 93.2% for Unclean Gear and 98.9% for Deburring; image-level accuracy reaches 100%	Demonstrate the effectiveness of preprocessing and augmentation in conditions of limited training data.[37]
Caballero-Ramírez et al. (2025)	Capsule detection on wine bottles	YOLOv8, YOLOv9, and YOLOv11	Precision >96%, recall >97%, and mAP >98%	Compare multiple generations of YOLO for packaging inspection with relatively large datasets and evaluate performance and inference speed.[38]
This Study	Packaging defect sorting (Good/Bad Products)	YOLOv8 + ESP32 + servo + infrared sensor + IoT (Blynk, Google Sheets)	Model accuracy 99.32%, mAP50 97.6% (no false negatives from 292 instances); physical sorting success 86% (Bad Products), 96% (Good Product); end-to-end response ~0.51s with 1.40s real-time margin	Closed-loop integration connecting YOLOv8 detection, physical actuation via servo/ESP32, and dual logging (LCD + IoT); validates both model-level and system-level performance with end-to-end latency characterization

The average end-to-end response time was 0.51 s (0.16 s inference, 0.05 s transmission, 0.30 s actuation), confirming real-time feasibility, while tests across five dataset split ratios showed that a larger training proportion does not necessarily improve performance. Overall, the study contributes a validated end-to-end system integrating YOLOv8, sensors, conveyor, ESP32, and IoT monitoring, highlighting the practical gap between detection accuracy and physical sorting performance.

4. CONCLUSION

This research successfully developed an automated packaging inspection and sorting system integrating YOLOv8, an ESP32, infrared sensors, a conveyor, and a servo. Testing showed the YOLOv8 model classified Good and Bad Products with strong performance, achieving 97.6% mAP@0.5 and precision/recall both above 98% at the detection-instance level. Across 292 evaluated instances, no false negatives occurred and only two false positives appeared, confirming the model's reliability on the test dataset.Deploying the model on the physical sorting system, however, revealed that model performance does not fully translate to system-level success. Over 100 trials, rejection success reached 86% for Bad Products and 96% for Good Products, indicating that overall performance depends not only on the detection model but also on mechanical and real-time factors such as communication delay, conveyor speed, product position, and servo response.

The core pipeline YOLOv8 inference, data transmission to the ESP32, and servo actuation averaged 0.51 seconds end-to-end, while monitoring via Blynk and logging via Google Sheets ran independently without hindering the sorting process, confirming the system's real-time feasibility under the tested laboratory conditions. This integration also enabled both local and remote monitoring and data logging, extending the study's contribution beyond YOLOv8's detection capability to the successful integration of computer vision, control systems, physical actuation, and IoT within a single automated sorting platform.Several limitations remain: a relatively small dataset (1,500 images), laboratory-only testing, use of only two classes, and reliance on a single camera angle. Future work should expand and diversify the dataset, validate the system in an industrial setting, introduce more specific Bad Products classes, incorporate multiple camera angles to reduce blind spots, and further optimize synchronization among YOLOv8 detection, the conveyor, ESP32, communication, and servo actuation to improve physical sorting success.

ACKNOWLEDGEMENTS

The completion of this research would not have been possible without the support of many individuals and institutions. The authors extend their deepest gratitude to the D4 Electrical Engineering Study Program, Universitas Negeri Surabaya, for providing the laboratory facilities, equipment, and academic environment that made this work possible. Special thanks are owed to As'ad Shidqy Aziz for the invaluable guidance, patience, and insightful feedback that shaped the direction of this study from its earliest stages to its completion.Finally, heartfelt appreciation goes to our families and friends, whose unwavering encouragement and understanding sustained us throughout this journey. This research was carried out independently, without financial support from any funding agency in the public, commercial, or not-for-profit sectors.

REFERENCES

[1] C. M. Badgujar, A. Poulose, and H. Gan, "Agricultural object detection with You Only Look Once (YOLO) Algorithm: A bibliometric and systematic literature review," *Comput. Electron. Agric.*, vol. 223, no. May, 2024, doi: 10.1016/j.compag.2024.109090.

[2] D. Saha, M. Padhiary, and N. Chandrakar, "AI vision and machine learning for enhanced automation in food industry: A systematic review," *Food Humanit.*, vol. 4, p. 100587, May 2025, doi: 10.1016/j.foohum.2025.100587.

[3] W. Xie *et al.*, "Towards high-accuracy digital detection in meter images: The ASS-YOLO algorithm and Dig21000 dataset," *Neurocomputing*, vol. 647, no. May, p. 130606, 2025, doi: 10.1016/j.neucom.2025.130606.

[4] H. Li, J. Huang, Z. Gu, D. He, J. Huang, and C. Wang, "Positioning of mango picking point using an improved YOLOv8 architecture with object detection and instance segmentation," *Biosyst. Eng.*, vol. 247, no. April, pp. 202–220, 2024, doi: 10.1016/j.biosystemseng.2024.09.015.

[5] X. Li, M. Hu, B. Li, and R. Tong, "OAM-YOLO: A real-time small object detection framework for PPE compliance monitoring in industrial environments," *Process Saf. Environ. Prot.*, vol. 204, no. May, p. 108058, 2025, doi: 10.1016/j.psep.2025.108058.

[6] H. Wang and H. Qian, "SDG-YOLOv8: Single-domain generalized object detection based on domain diversity in traffic road scenes," *Displays*, vol. 87, no. January, p. 102948, 2025, doi: 10.1016/j.displa.2024.102948.

[7] Y. Chen *et al.*, "Guided feature learning network focusing on spatial and frequency domains for real-time small object detection," *Eng. Appl. Artif. Intell.*, vol. 181, no. P4, p. 115386, 2026, doi: 10.1016/j.engappai.2026.115386.

[8] L. J. Liu, S. Q. Sun, and H. R. Karimi, "A real-time surface defect detection model based on adaptive feature information selection and fusion," *Inf. Fusion*, vol. 129, no. June 2025, p. 104041, 2026, doi: 10.1016/j.inffus.2025.104041.

[9] R. Ranjan, "YOLO in precision aquaculture: A decadal bibliometric and systematic review of applications, architectural adaptations, and deployment challenges," *J. Agric. Food Res.*, vol. 28, p. 102982, 2026, doi: 10.1016/j.jafr.2026.102982.

[10] C. C. Kuo *et al.*, "SE-ATT-YOLO- A deep learning driven ultrasound based respiratory motion compensation system for precision radiotherapy," *Comput. Biol. Med.*, vol. 195, no. June, p. 110638, 2025, doi: 10.1016/j.compbiomed.2025.110638.

[11] S. Kumari *et al.*, "YOLO-DwSimAM: An improved YOLOv11 network for insect-pest detection in rice crop," *Neurocomputing*, vol. 698, no. June, p. 134315, 2026, doi: 10.1016/j.neucom.2026.134315.

[12] B. Ma *et al.*, "Using an improved lightweight YOLOv8 model for real-time detection of multi-stage apple fruit in complex orchard environments," *Artif. Intell. Agric.*, vol. 11, pp. 70–82, 2024, doi: 10.1016/j.aiia.2024.02.001.

[13] A. Martikkala, J. David, A. Lobov, M. Lanz, and I. F. Ituarte, "Trends for low-cost and open-source IoT solutions development for industry 4.0," *Procedia Manuf.*, vol. 55, no. C, pp. 298–305, 2021, doi: 10.1016/j.promfg.2021.10.042.

[14] V. Moya, M. Guerra, K. Pazmiño, F. Abedrabbo, F. A. Chicaiza, and D. Pozo-Espín, "Tomato classification with YOLOv8: Enhancing automated sorting and quality assessment," *Smart Agric. Technol.*, vol. 12, no. August, p. 101221, 2025, doi: 10.1016/j.atech.2025.101221.

[15] Y. M. Abdal, A. H. Ali, and O. N. M. Salim, "IoT-enabled monitoring, fuzzy control, and LSTM-based temperature forecasting for smart hydroponic greenhouses," *Smart Agric. Technol.*, vol. 15, no. May, p. 102432, 2026, doi: 10.1016/j.atech.2026.102432.

[16] Belinda Ayuningtyas, Riesa Syariful Akbar, Syaeful Ilman, R. Rustandi, and N. Nurudin, "Prototype of an IoT-Based Height Sorting Conveyor using ESP32, HC-SR04, and IR Sensors," *J. Ilm. Tek.*, vol. 5, no. 1, pp. 223–237, 2026, doi: 10.56127/juit.v5i1.2532.

[17] V. Lakshmi Chetana, V. Srilakshmi, and S. Arukonda, "YOLOv8-Based Real-Time Traffic Sign Detection for Intelligent Transportation Systems," *Procedia Comput. Sci.*, vol. 283, pp. 1876–1886, 2026, doi: 10.1016/j.procs.2026.06.262.

[18] P. Dong, Y. Wang, Q. Yu, W. Feng, and G. Zong, "AMC-YOLO: Improved YOLOv8-based defect detection for cigarette packs," *IET Image Process.*, vol. 18, no. 14, pp. 4873–4886, 2024, doi: 10.1049/ipr2.13272.

[19] F. Pan, J. Li, Y. Yan, S. Guan, B. Biswal, and Y. Zhao, "Enhanced surface defect detection of cylinder liners using Swin Transformer and YOLOv8," *J. Autom. Intell.*, vol. 4, no. 3, pp. 227–235, 2025, doi: 10.1016/j.jai.2025.01.004.

[20] C. Agnew, E. M. Grua, P. Van de Ven, P. Denny, C. Eising, and A. Scanlan, "Pretraining instance segmentation models with bounding box annotations," *Intell. Syst. with Appl.*, vol. 24, no. September, p. 200454, 2024, doi: 10.1016/j.iswa.2024.200454.

[21] M. Chaman, A. El Maliki, H. Dahou, and A. Hadjoudja, "Benchmarking YOLO-based deep learning models for real-time object detection in hybrid ADAS and intelligent transportation systems," *Results Eng.*, vol. 29, no. December 2025, p. 108942, 2026, doi: 10.1016/j.rineng.2025.108942.

[22] A. A. Murat and M. S. Kiran, "A comprehensive review on YOLO versions for object detection," *Eng. Sci. Technol. an Int. J.*, vol. 70, no. January, 2025, doi: 10.1016/j.jestch.2025.102161.

[23] W. Luo and S. Yuan, "Enhanced YOLOv8 for small-object detection in multiscale UAV imagery: Innovations in detection accuracy and efficiency," *Digit. Signal Process. A Rev. J.*, vol. 158, no. December 2024, p. 104964, 2025, doi: 10.1016/j.dsp.2024.104964.

[24] P. Vasanthi and L. Mohan, "Efficient YOLOv8 algorithm for extreme small-scale object detection," *Digit. Signal Process. A Rev. J.*, vol. 154, no. July, p. 104682, 2024, doi: 10.1016/j.dsp.2024.104682.

[25] L. Malburg, M. P. Rieder, R. Seiger, P. Klein, and R. Bergmann, "Object detection for smart factory processes by machine learning," *Procedia Comput. Sci.*, vol. 184, no. 2019, pp. 581–588, 2021, doi: 10.1016/j.procs.2021.04.009.

[26] B. I. Oladapo *et al.*, "Model design and simulation of automatic sorting machine using proximity sensor," *Eng. Sci. Technol. an Int. J.*, vol. 19, no. 3, pp. 1452–1456, 2016, doi: 10.1016/j.jestch.2016.04.007.

[27] W. Tan, Q. Duan, L. Yao, and J. Li, "A sensor combination based automatic sorting system for waste washing machine parts," *Resour. Conserv. Recycl.*, vol. 181, no. November 2021, p. 106270, 2022, doi: 10.1016/j.resconrec.2022.106270.

[28] P. Dera, T. Talaśka, and R. Długosz, "A new, cost-efficient modular sensor platform for IoT and predictive maintenance in industrial applications," *J. Comput. Appl. Math.*, vol. 473, no. December 2024, p. 116777, 2026, doi: 10.1016/j.cam.2025.116777.

[29] A. Mumuni and F. Mumuni, "Data augmentation: A comprehensive survey of modern approaches," *Array*, vol. 16, no. November, p. 100258, 2022, doi: 10.1016/j.array.2022.100258.

[30] M. F. A. Sayeedi, A. M. I. M. Osmani, T. Rahman, J. F. Deepti, R. Rahman, and S. Islam, "ElectroCom61: A multiclass dataset for detection of electronic components," *Data Br.*, vol. 59, p. 111331, 2025, doi: 10.1016/j.dib.2025.111331.

[31] M. J. A. Rafi, M. Rony, and N. Majadi, "NSTU-BDTAKA: An open dataset for Bangladeshi paper currency detection and recognition," *Data Br.*, vol. 55, p. 110701, 2024, doi: 10.1016/j.dib.2024.110701.

[32] H. Assemlali, S. Bouhsissin, and N. Sael, "Deep learning-driven CNN model for detection and classification of dynamic obstacles," *Green Energy Intell. Transp.*, vol. 5, no. 3, p. 100334, 2026, doi: 10.1016/j.geits.2025.100334.

[33] C. Newman, J. Petzing, Y. M. Goh, and L. Justham, "Investigating the optimisation of real-world and synthetic object detection training datasets through the consideration of environmental and simulation factors," *Intell. Syst. with Appl.*, vol. 14, 2022, doi: 10.1016/j.iswa.2022.200079.

[34] S. Pe *et al.*, "How improper dataset split hinders model generalizability: a systematic comparison in Human activity recognition and exercise evaluation tasks," *Int. J. Med. Inform.*, vol. 216, no. April, p. 106464, 2026, doi: 10.1016/j.ijmedinf.2026.106464.

[35] M. Vilares Ferro, Y. Doval Mosquera, F. J. Ribadas Pena, and V. M. Darriba Bilbao, "Early stopping by correlating online indicators in neural networks," *Neural Networks*, vol. 159, pp. 109–124, 2023, doi: 10.1016/j.neunet.2022.11.035.

[36] R. Sapkota, D. Ahmed, and M. Karkee, "Comparing YOLOv8 and Mask R-CNN for instance segmentation in complex orchard environments," *Artif. Intell. Agric.*, vol. 13, pp. 84–99, 2024, doi: 10.1016/j.aiia.2024.07.001.

[37] X. Zhu, M. Björkman, A. Maki, L. Hanson, and P. Mårtensson, "Surface Defect Detection with Limited Training Data: A Case Study on Crown Wheel Surface Inspection," *Procedia CIRP*, vol. 120, pp.

ISSN: 2715-6427

1333–1338, 2023, doi: 10.1016/j.procir.2023.09.172.
[38] D. Caballero-Ramírez *et al.*, "Deep learning-based detection of wine bottle capsules for contamination prevention," *Results Eng.*, vol. 27, no. March, 2025, doi: 10.1016/j.rineng.2025.106388.