



Design and Implementation of a Real-Time Waste Detection and Sorting System Using YOLOv8n-Based Embedded Edge AI

Anh-Quan Dao¹, Xuan-Minh Dinh², Van-Dung Le¹, Minh-Tri Nguyen Sy³, Tien-Ha Nguyen¹,
Van-Tuan Nguyen^{2*}

¹ Faculty of Engineering and Technology, Hung Vuong University, Phu Tho Province 35000, Vietnam

² Faculty of Mechanical Engineering and Mechatronics, Phenikaa University, Duong Noi, Hanoi 12116, Vietnam

³ HTI UAS Joint Stock Company, HTI Group, Hanoi 12016, Vietnam

Corresponding Author Email: tuan.nguyenvan@phenikaa-uni.edu.vn

Copyright: ©2026 The authors. This article is published by IETA and is licensed under the CC BY 4.0 license (<http://creativecommons.org/licenses/by/4.0/>).

<https://doi.org/10.18280/jesa.590819>

ABSTRACT

Received: 4 June 2026

Revised: 14 August 2026

Accepted: 23 August 2026

Available online: 31 August 2026

Keywords:

You Only Look Once, deep learning, embedded systems, edge AI, waste detection, automatic waste sorting systems

Waste management and sorting play an important role in reducing environmental pollution and improving recycling efficiency. However, conventional sorting methods rely largely on manual labor, resulting in limited efficiency and poor scalability. In addition, current cloud-computing-based solutions often involve relatively high deployment and maintenance costs. This research proposes a waste-detecting and mechanical sorting system based on the YOLOv8n model to address these issues. This study's primary contribution is not a new You Only Look Once (YOLO) architecture, but rather the design, integration, and assessment of an entire edge AI system that takes into account the detection model, edge computing device, controller, and actuator mechanism simultaneously. A custom-built dataset comprising 4,000 images from four waste categories—paper, metal, plastic, and other waste—was used to train and evaluate the models. The images were collected under various lighting conditions, backgrounds, distances, viewing angles, and object orientations. Rotation and flipping operations were also applied to increase data diversity. Three lightweight object detection models, namely YOLOv5n, YOLOv7-tiny, and YOLOv8n, were trained and compared under the same conditions. After 100 training epochs, the best YOLOv8n checkpoint achieved a Precision of 0.847, a Recall of 0.805, an mAP@0.5 of 0.888, an mAP@0.5:0.95 of 0.728, and an F1-score of 0.82 at a confidence threshold of 0.512. The model was deployed on a Raspberry Pi 5 (RP5) to process images captured by a camera and transmit the detection results to an Arduino Uno, which controlled a stepper motor through a TB6600 driver and a servo motor to direct each waste item into the corresponding bin. The deployment results showed that the system achieved a processing speed of 6.69 ± 0.43 frames per second (FPS), an inference latency of 150.47 ± 15.33 ms, a central processing unit (CPU) utilization of $54.61 \pm 4.34\%$, a RAM usage of 790.72 ± 1.68 MB, and a power consumption of approximately 19.14 W. During 150 consecutive operating trials, the system successfully completed 137 trials, corresponding to a success rate of 91.3%. These results demonstrate the feasibility of deploying the lightweight YOLOv8n model on an edge computing platform for waste detection and mechanical sorting applications.

1. INTRODUCTION

Nowadays, on a global scale, population growth together with urbanization, industrialization, and resource exploitation activities has significantly increased the amount of generated waste, becoming one of the major causes of environmental pollution and posing serious threats to human health if not properly managed and treated [1]. Traditional methods such as visual inspection and manual sorting, although widely adopted, still suffer from several limitations due to their subjectivity and dependence on workers' experience, leading to frequent errors, low efficiency, and poor scalability in handling large volumes of waste in real-world applications [2]. In addition, manual waste sorting methods require workers to

be regularly exposed to polluted environments and hazardous substances, thereby increasing risks to occupational health and safety. In response to these challenges, the development of intelligent waste classification solutions based on AI has become increasingly essential, contributing to the reduction of dependence on manual labor while improving waste management efficiency.

Machine learning (ML) enables computers to learn from data in order to make decisions; however, traditional ML methods face limitations in processing raw natural data [3]. Deep learning (DL), as a branch of ML, provides the capability to learn directly from raw data without requiring extensive and complex preprocessing steps [4]. DL-based object detection methods in computer vision are generally classified into two

main groups: two-stage detectors and one-stage detectors [5]. Among modern object detection frameworks, You Only Look Once (YOLO) [6], which belongs to the family of one-stage object detection algorithms, has become one of the most widely used methods for real-time waste detection applications due to its balance between speed and accuracy. Unlike two-stage methods such as Faster Region-based Convolutional Neural Network (Faster R-CNN) [7], which first perform localization and then classification, one-stage algorithms directly extract features within the network to simultaneously predict object locations and classifications. As a result, YOLO models have a simpler architecture and significantly improved detection speed, although with a slight reduction in accuracy. However, the improvements introduced in recent YOLO versions, particularly YOLOv8 [8], which was released as an open-source framework by Ultralytics in 2023, have substantially enhanced both processing speed and object detection accuracy. Thanks to a more efficient backbone architecture compared to previous versions, YOLOv8 achieves lower computational cost while maintaining high detection accuracy, thereby narrowing the accuracy gap with two-stage detection methods.

The YOLOv8 family is available in five model scales: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x. These versions provide different trade-offs among detection performance, model size, and computational demand [9]. Among them, YOLOv8n is the lightest version, optimized for edge computing applications and hardware platforms with limited computational resources, while still maintaining suitable object detection performance for real-time applications. In addition, YOLOv8 is designed to facilitate easy switching between different YOLO versions and to be compatible with various hardware platforms, including both central processing unit (CPU) and graphics processing unit (GPU) [10]. Castro-Bello et al. [11] demonstrated that YOLOv8n achieves more accurate and stable waste detection performance compared to YOLOv4-tiny and YOLOv7-tiny models. In contrast, YOLOv4-tiny is highly dependent on lighting conditions of waste objects, whereas YOLOv7-tiny exhibits high CPU utilization, requires close observation distances, and in some cases produces empty bounding boxes during detection. Meanwhile, Okano et al. [12] trained YOLOv8n and YOLOv8s models on a GPU server and deployed them on a Raspberry Pi 5 (RP5) using only the CPU. The obtained results showed that YOLOv8s requires longer inference time due to its higher computational complexity, whereas YOLOv8n achieves faster processing speed with stable performance. Furthermore, the results reported in reference [13] indicate that YOLOv8n provides higher processing speed with only a slight reduction in accuracy compared to YOLOv8x, thereby demonstrating the suitability of YOLOv8n for applications prioritizing real-time processing capability.

In addition, many recent studies have also implemented YOLO-based approaches for waste detection and achieved remarkable results [14-20]. Li et al. [14] improved YOLOv8 by employing CG-HGNetV2 as the backbone and integrating an attention module named MSE-AKConv, thereby reducing the computational cost of the model and improving the accuracy of locating and identifying large waste objects. Liu et al. [15] proposed EcoDetect-YOLO based on the YOLOv5s framework, which effectively detects waste, particularly paper and plastic waste. Dipo et al. [16] implemented the YOLOv12 model for waste detection and classification, where the

proposed system achieved an impressive accuracy of 73% and a mean Average Precision (mAP) of 78%. These results demonstrate that YOLO-based architectures remain a highly promising approach for intelligent waste classification systems. However, the studies analyzed above mainly focus on improving model performance and evaluating the models on hardware platforms with abundant resources and strong computational capabilities. Practical deployment and evaluation on edge computing devices with limited hardware resources have not yet been comprehensively investigated, especially in the current context where intelligent waste management systems integrating DL models on edge devices with Internet of Things (IoT) infrastructure are attracting increasing attention [21].

Rahmatulloh et al. [17] investigated the use of the YOLOv7 algorithm for waste detection and classification. The methodology comprised dataset collection, data preprocessing, model development and training, followed by evaluation on the test set. The reported results showed a Precision (P) of 0.801, an mAP@0.5 of 0.868, and an mAP@0.5:0.95 of 0.618. These results demonstrated the capability of YOLOv7 to detect waste objects in the evaluated dataset. However, the study was confined to model-level assessment and did not examine deployment on embedded computing platforms. In addition, practical deployment metrics, including inference speed, computational resource usage, and power consumption, were not reported. Such factors are important for assessing the suitability of an object detector for real-time waste sorting systems.

Partosan et al. [18] deployed the YOLOv8 model on a RP5 integrated with a Raspberry Pi Camera v3 to detect five waste categories within a school campus environment. The authors used a dataset comprising 3,718 images with a resolution of 640×640 pixels and evaluated the YOLOv8n, YOLOv8s, YOLOv8m, and YOLOv8l variants. The results showed that YOLOv8n achieved the best performance among the evaluated variants while providing a strong trade-off between detection performance and computational cost, supporting its use in edge AI applications. However, the study mainly focused on evaluating the model's detection metrics and its deployability on the RP5. Edge-device operating parameters, such as CPU utilization, RAM usage, inference latency, and processing speed, were not analyzed in detail. In addition, the proposed setup only included image acquisition and waste detection modules, without integrating an actuation mechanism to automatically direct each waste category into the corresponding bin. The cycle time and continuous-operation capability of the overall system were also not evaluated. Therefore, the study did not constitute a complete waste detection and mechanical sorting system.

Siyad et al. [19] developed a complete automated waste sorting system by integrating a YOLOv11n model deployed on an RP5 with a two-degree-of-freedom actuation mechanism driven by stepper motors. The model detected 19 object classes, which were subsequently mapped into four main waste categories: biodegradable waste, plastic, metal, and hazardous waste. The authors constructed a dataset comprising 55,134 images containing 106,044 labeled objects, and the model achieved an mAP@0.5 of 90.17% on the test set. During the edge-deployment evaluation, the study showed that the YOLOv11x variant achieved only approximately 1-3 frames per second (FPS) when executed directly on the RP5, which was insufficient to meet the real-time requirements of the dynamic sorting mechanism. Therefore, YOLOv11n was

selected for deployment on the RP5 CPU. However, the paper did not report the actual inference speed of YOLOv11n. The study also evaluated several operational parameters of the overall system, including power consumption, which ranged from 27.4 to 30 W. Nevertheless, CPU utilization and RAM usage on the RP5 during inference were not reported. In addition, the sorting cycle time was presented only as an overall average and was not analyzed separately for each waste category. Although the prototype was evaluated for operational stability over a 21-day period, the results were not quantified in terms of the total number of sorting trials, sorting success rate, or mechanical failures that could occur during operation.

Echeverry and López [20] proposed AUTORECYCLER, a computer-vision-based prototype for the automated sorting of recyclable waste. The system deployed an SSD-MobileNet-v2 model on a Jetson Nano platform and used a USB camera to acquire input images in real time. The experimental results showed that the system achieved high sorting accuracy. However, the study still had several limitations, including the lack of power-consumption measurements using dedicated metering equipment and the use of a chute-based sorting mechanism, in which object motion could be affected by shape characteristics and contact conditions. Improving this aspect was also identified by the authors as a direction for future work.

An alternative approach for intelligent waste management was proposed in reference [22], in which the authors combined a genetic algorithm (GA) with a fuzzy inference system (FIS). Specifically, the GA was employed to optimize the rule set of a Mamdani fuzzy inference model in order to determine the most suitable rule configuration for the system. Experimental results demonstrated that the GA-FIS model achieved high performance in determining the status of smart waste bins, with overall accuracy and Recall (R) values of 96.68% and 93.96%, respectively, when aggregating results across all classes. In addition, the system showed relatively effective recognition capability for several waste categories such as paper, metal, and plastic. However, fuzzy inference-based methods generally depend heavily on the careful design of rule sets and membership functions, which requires significant expertise from the system designer. This dependency makes such approaches difficult to scale when the number of object classes or the complexity of the data increases, and their performance may degrade if the fuzzy inference rules are not properly tuned [23]. Furthermore, GA-FIS models still exhibit several limitations compared to modern DL approaches such as YOLO, particularly in real-time object detection tasks, the ability to process complex image data, and scalability as well as integration into large-scale intelligent systems.

Inspired by the aforementioned studies and motivated by the identified research gaps, this study focuses on the design and implementation of a waste detection and classification system based on the YOLOv8n model. Although many next-generation YOLO versions have recently emerged, such as YOLO26 [24], YOLOv8 is still highly regarded by the research community due to its proven stability and its ability to achieve a strong balance between accuracy, processing speed, and practical deployment capability on resource-constrained hardware platforms. In particular, the YOLOv8n variant features a small number of parameters and low computational cost, while also being strongly supported through the Ultralytics framework, facilitating training, optimization, and deployment on edge AI devices. Compared

with widely used YOLO architectures such as YOLOv5 and YOLOv7, YOLOv8 provides higher detection accuracy and faster processing speed [25]. In addition, the use of the C2f module enhances information flow during feature extraction through a gradient branch connection mechanism while still maintaining the compact and efficient architecture of the model. Furthermore, the Anchor-Free mechanism together with the multi-scale feature fusion structure, feature pyramid network-path aggregation network (PAN-FPN) enables YOLOv8 to reduce computational complexity and improve the detection performance of objects with different sizes within the same image. Moreover, the combination of complete intersection over union (CIoU) [26] and distribution focal loss (DFL) [27] for bounding box regression, along with binary cross-entropy (BCE) for classification tasks, helps YOLOv8 improve localization capability and the detection performance of small objects [28]. This makes the model particularly suitable for real-world waste detection and classification applications. In addition, YOLO-based models generally reduce the dependence on expert knowledge compared to fuzzy logic-based approaches such as the method presented in reference [22].

The paper presents the complete workflow, encompassing dataset construction, model training and performance evaluation, experimental hardware design, and system deployment and evaluation under real-world operating conditions. In this study, model training and evaluation were conducted using the Ultralytics framework on Google Colab with an NVIDIA Tesla T4 GPU. A diverse dataset comprising 4,000 images from four waste categories—paper, metal, plastic, and other waste—was constructed for model training. The images were collected under varying lighting conditions, backgrounds, distances, viewing angles, and object orientations. Rotation and flipping operations were applied to increase data diversity and support model generalization. Using this dataset, three lightweight object detection models, namely YOLOv5n, YOLOv7-tiny, and YOLOv8n, were trained and evaluated under the same conditions. The results showed that YOLOv8n achieved a P of 0.847, a R of 0.805, an mAP@0.5 of 0.888, an mAP@0.5:0.95 of 0.728, and an F1-score (F1) of 0.82. The model achieved the highest P and mAP@0.5:0.95 while maintaining mAP@0.5 and F1 values comparable to those of YOLOv5n. Therefore, YOLOv8n was selected for deployment in the proposed system.

In addition to model evaluation, this study designed and implemented a complete integrated waste detection and mechanical sorting system. In the proposed system, a RP5 with 8 GB of RAM acquires images from a camera and performs waste detection using the YOLOv8n model. The detection results are transmitted to an Arduino Uno, which controls a stepper motor through a TB6600 driver and a servo motor to direct each object into the corresponding bin. The obtained results show that the proposed prototype is capable of detecting waste and directing it to the correct bin under the evaluated real-world operating conditions, demonstrating the feasibility of deploying lightweight DL models such as YOLOv8n on embedded edge computing platforms.

The system's deployability on the edge device was evaluated from multiple perspectives. In addition to the P, R, F1, and mAP metrics, the study analyzed inference latency, processing speed, CPU utilization, RAM usage, and power consumption. The operation of the sorting mechanism was also evaluated based on the cycle completion time for each waste category and the continuous-operation capability of the

overall system. The results showed that the system achieved a processing speed of 6.69 ± 0.43 FPS, an inference latency of 150.47 ± 15.33 ms, a CPU utilization of $54.61 \pm 4.34\%$, a RAM usage of 790.72 ± 1.68 MB, and a power consumption of approximately 19.14 W. During 150 consecutive operating trials, the system successfully completed 137 trials, corresponding to a success rate of 91.3%; unsuccessful trials were analyzed in terms of detection errors and mechanical failures.

Therefore, the key contribution of this study does not lie in proposing a new YOLO architecture, but rather in the design, integration, and evaluation of a complete edge AI system for waste detection and mechanical sorting. In this system, model performance, computational resource usage on the RP5, power consumption, actuator operation, sorting cycle time, and continuous-operation capability are evaluated simultaneously. This approach extends the scope of assessment from the object detection model level to the overall system level, thereby providing practical evidence for evaluating the deployment feasibility of lightweight DL models in smart waste management applications.

In addition, the results obtained from this study can serve as a valuable reference source for engineers, researchers, and developers in the design and deployment of edge AI systems for intelligent waste management applications, as well as for automation problems involving the integration of computer vision and robotics, similar to the system proposed in reference [29].

The paper is structured into four main sections. Section 2 describes the proposed methodology, covering the system architecture, operational workflow, YOLOv8n model, and experimental hardware configuration. Section 3 provides detailed experimental results and analysis. Finally, Section 4 presents the conclusions and discusses future research directions.

2. PROPOSED METHODOLOGY

2.1 System architecture and workflow

Figure 1 illustrates the overall architecture of the proposed waste detection and classification system, which consists of five main components: (i) the image acquisition module, (ii) the processing and inference module, (iii) the control module, (iv) the power supply system, and (v) the actuation system. In particular, the image acquisition module uses an external camera to continuously acquire visual data from the waste objects presented to the system. The captured image data are then transmitted to the processing and inference module, which serves as the CPU of the system. This module employs an RP5 embedded computer running the YOLOv8n model to detect four waste categories: paper, metal, plastic, and other waste. The output of the model includes bounding boxes, class labels, and corresponding confidence scores. The control module uses an Arduino Uno microcontroller to receive the classification results from the Raspberry Pi through Serial Communication. Based on these results, the Arduino Uno generates control signals for the actuation mechanisms, including motor drivers and motors, in order to perform the automatic waste sorting process. The actuation system is responsible for converting electrical signals into mechanical motion in order to direct waste to the corresponding sorting location. The power supply system is designed with a separate architecture to ensure safety and stable operation of the entire system. Specifically, the 220 VAC mains power is converted into a 5 VDC supply for the RP5, Arduino Uno, and servo motor, while a 24 VDC power supply is used for the TB6600 driver and the stepper motor. The separation between the 5 VDC supply and the 24 VDC power supply helps ensure that each device operates at its rated voltage, thereby improving system stability, enhancing operational safety, and extending the service life of the electronic hardware components.

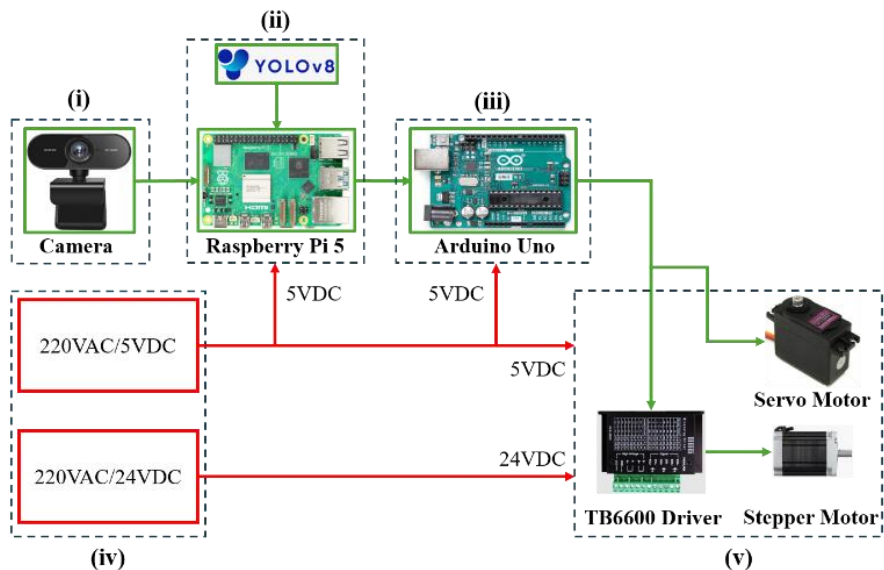


Figure 1. Architecture of the proposed waste detection and sorting system

Figure 2 presents the overall workflow of the proposed YOLOv8n-based waste detection and sorting system. The procedure consists of two main phases: the construction and training stage of the YOLOv8n model, and the experimental deployment stage of the waste detection and sorting system.

In the first stage, waste data are collected and organized into

a complete dataset. The dataset is then divided into three subsets, including the training set, Validation Set, and Testing Set, with corresponding ratios of 80%, 10%, and 10%, respectively. To improve the generalization capability of the model, the training set is further expanded using data augmentation techniques, forming an extended training set.

This dataset is then used to train the YOLOv8n model in order to optimize its ability to detect waste objects across the four target categories. During the training process, the Validation Set is used to monitor model performance and reduce the risk of overfitting. After training is completed, the model is evaluated using the Testing Set to obtain performance metrics such as P, R, AP, mAP, and F1.

In the second stage, the trained YOLOv8n model is implemented on the RP5 to process camera images and detect waste objects in real time. After an object is detected and its class label is determined, the RP5 transmits the detection result to the Arduino Uno via serial communication. The Arduino Uno then generates control signals for the actuation mechanism, which directs the object to the corresponding waste bin. Through this sequential integration of visual detection, embedded control, and mechanical actuation, the proposed system performs automatic waste detection and sorting.

2.2 Waste detection model based on YOLOv8n

YOLO-based models, in general, have achieved remarkable results in the field of computer vision and demonstrated high applicability in various automation problems. In this study, the YOLOv8n architecture was selected for the waste detection task due to its lightweight design, fast inference capability, high detection accuracy, and suitability for edge AI platforms

with limited hardware resources. Compared with previous YOLO versions widely used in industry, such as YOLOv5 and YOLOv7, YOLOv8 demonstrates significant improvements in both detection accuracy and processing speed.

The overall architecture of the YOLOv8 model, mainly referenced from studies [25, 30-32], is illustrated in Figure 3, while the detailed network architecture is presented in Figure 4. The YOLOv8 model consists of three main components: Backbone, Neck, and Head [25]. Regarding the loss function, YOLOv8 employs a combination of CIoU [26] and DFL [27] for bounding box regression, together with BCE for classification tasks, in order to improve object detection performance, particularly for small objects [28].

Backbone: The backbone architecture of YOLOv8 is largely similar to that of YOLOv5; however, YOLOv8 replaces the C3 module used in YOLOv5 with the C2f module [31]. The backbone receives an input image of size $640 \times 640 \times 3$ and performs feature extraction through multiple convolutional layers (ConvModule) combined with C2f modules. During this process, the input features are progressively downsampled through five stages to produce five feature levels of different scales, denoted as P1-P5. The C2f module employs a gradient branching connection mechanism to enhance information flow during feature extraction while maintaining a lightweight and efficient architecture [25].

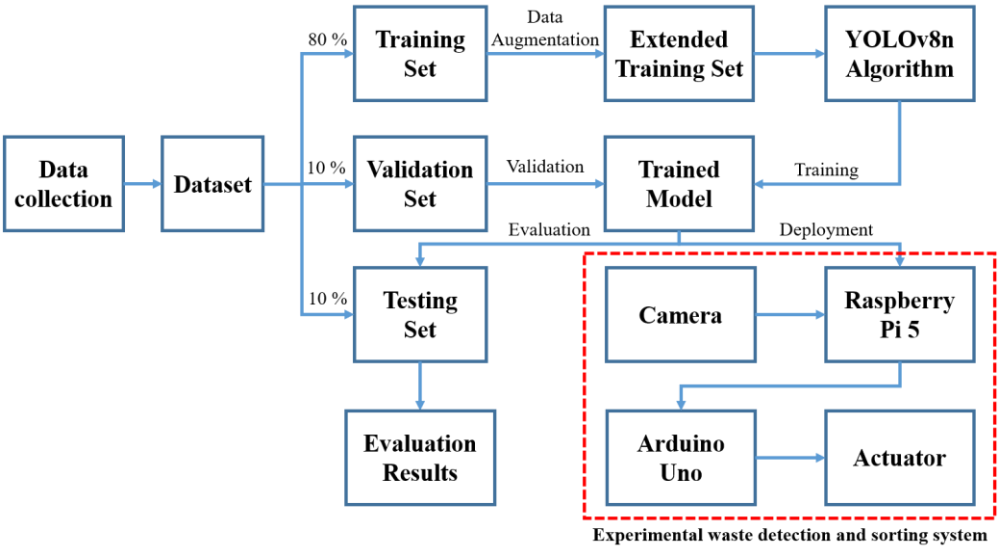


Figure 2. Overall workflow of the proposed real-time waste detection and sorting system based on YOLOv8n

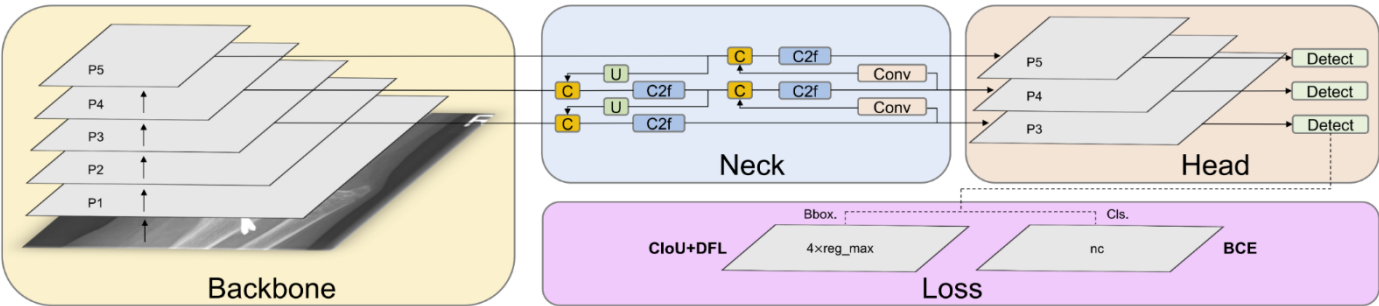


Figure 3. Illustration of the general architecture of the YOLOv8 model [30]

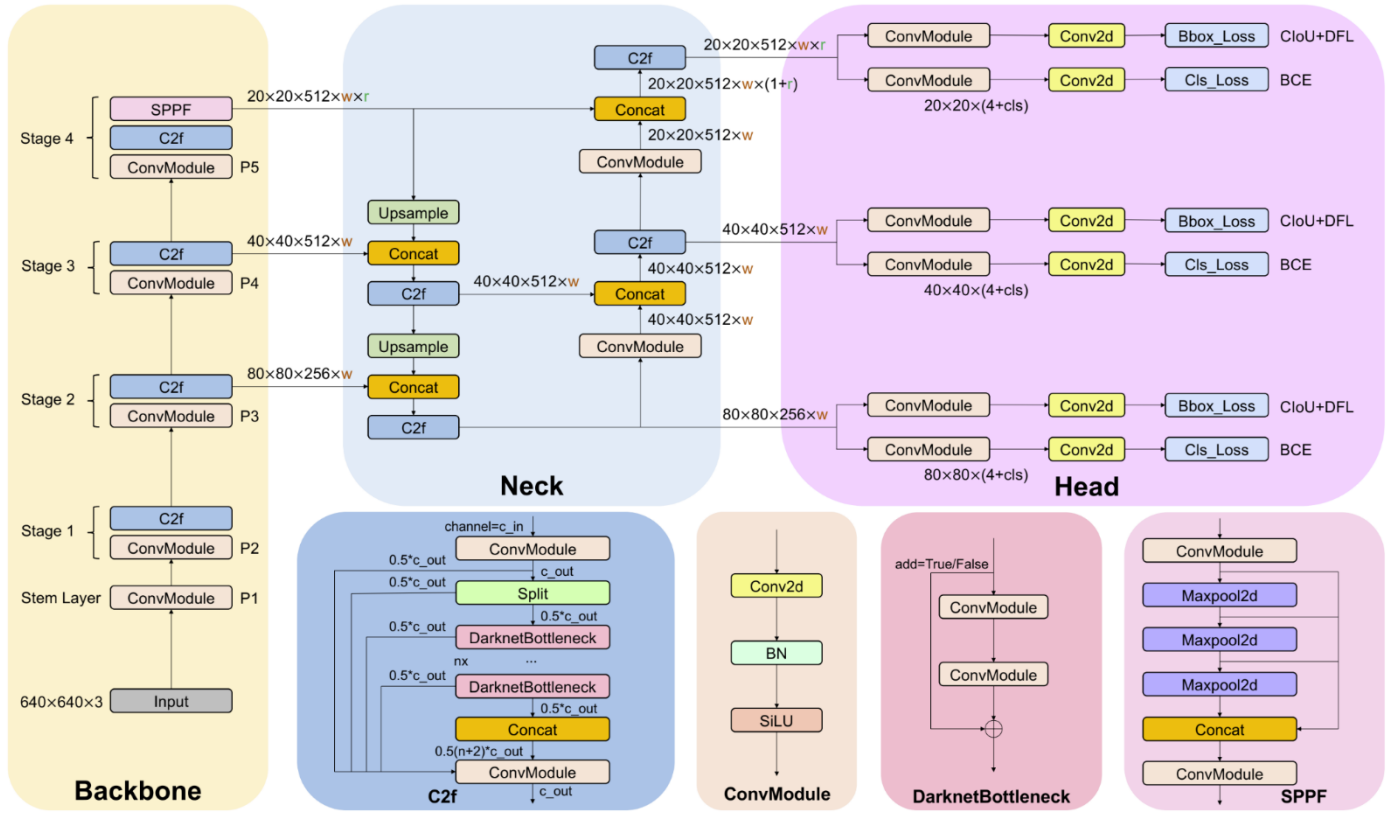


Figure 4. Illustration of the detailed architecture of the YOLOv8 model [30]

Specifically, as illustrated in Figure 4, C2f consists of two ConvModules and n Darknet bottlenecks connected via Split and Concat operations. Each ConvModule includes Conv-BN-SiLU components, where n denotes the number of bottlenecks [30]. In addition, the spatial pyramid pooling fast (SPPF) module is integrated at Stage 4 to aggregate input feature maps into a fixed size, thereby enabling outputs that are adaptable to objects of varying scales. Compared with the traditional spatial pyramid pooling (SPP), SPPF reduces computational cost and latency by using a sequential structure of three MaxPool layers, as shown in Figure 4 [25]. These improvements enable YOLOv8 to enhance feature learning capability while reducing computational cost and inference time.

Neck: As illustrated in Figure 4, the Neck connects the Backbone to the detection Head and performs multi-scale feature fusion. It combines the top-down pathway of the FPN with the bottom-up aggregation pathway of the PAN, while eliminating convolution operations during the feature upsampling stage to reduce computational cost. This architecture enables the effective integration of features extracted from different layers of the network. Specifically, FPN performs top-down feature upsampling to enrich low-level feature maps with high-level semantic information. In contrast, PAN conducts bottom-up feature aggregation through downsampling, thereby enhancing localization information in higher-level feature maps. These two feature flows are subsequently fused to improve the model's ability to accurately detect objects of varying sizes [32].

Head: YOLOv8 employs a decoupled head architecture, in which object classification and bounding box regression are performed by two independent branches [32]. This design improves both accuracy and convergence during training compared to the coupled head architecture used in YOLOv5. For the loss function, YOLOv8 combines CIoU and DFL for

bounding box regression, while BCE is adopted for classification, thereby enhancing object detection performance, particularly for small objects. In addition, YOLOv8 incorporates an anchor-free detection mechanism, which simplifies the object detection process, reduces computational complexity, and improves overall model efficiency [30]. Predictions are generated at three feature scales: P3 (80×80) for small objects, P4 (40×40) for medium-sized objects, and P5 (20×20) for large objects. This multi-scale detection mechanism enables the network to accurately detect objects of varying sizes within the same image.

2.3 Hardware design and experimental implementation

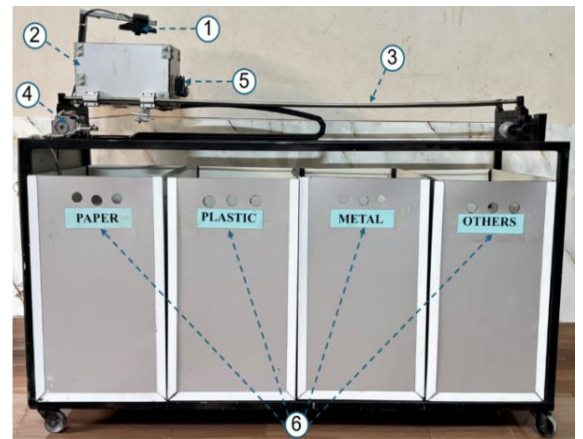


Figure 5. Experimental prototype of the waste detection and sorting system

Figure 5 illustrates the experimental prototype of the waste

detection and sorting system developed in this study. The numbered components of the prototype and their corresponding functions are presented in Table 1. The hardware and software configuration of the system is summarized in Table 2, while the system connection and wiring diagram is shown in Figure 6.

In the experimental system, the camera is directly connected to the RP5 to acquire input images for the waste detection and sorting process. The RP5 performs inference using the YOLOv8n model and transmits the detection results to the Arduino Uno via serial communication. The Arduino Uno serves as an intermediate controller, generating control signals for the TB6600 driver, stepper motor, and MG996R servo motor to execute the automatic waste sorting mechanism. In addition, the system uses two independent power supplies,

rated at 5 VDC and 24 VDC, to provide the appropriate operating voltage for each device and ensure stable operation of the overall system.

The connection diagram among the camera, RP5, Arduino Uno, TB6600 driver, and actuators is presented in Figure 6. The camera transmits images directly to the RP5, while the detection results are sent to the Arduino Uno via serial communication. The Arduino Uno generates control signals for the TB6600 driver to operate the stepper motor, moving the receiving box to the corresponding sorting position, while simultaneously controlling the servo motor to open and close the bottom flap. After the object is discharged into the designated bin, the stepper motor is driven by a predefined number of steps to return the receiving box to its initial position.

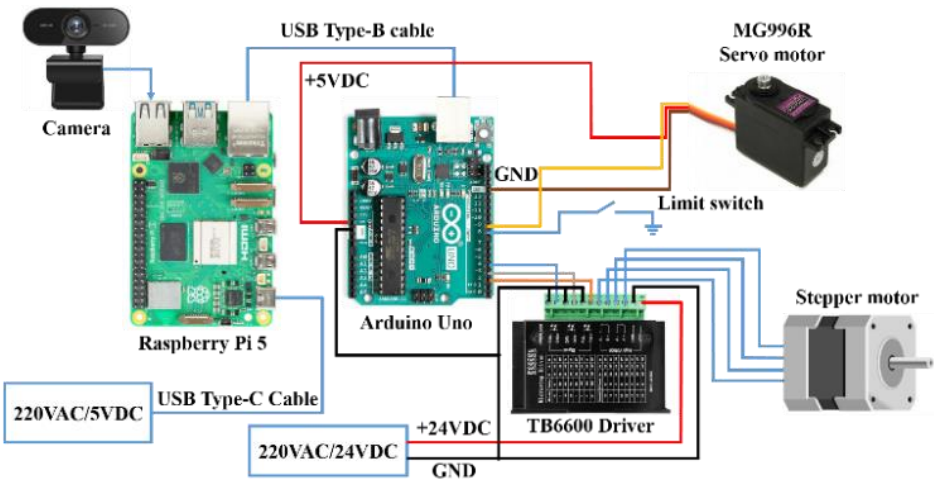


Figure 6. Wiring and connection diagram of the waste detection and sorting system

Table 1. The numbered components in Figure 5 and their corresponding functions

No.	Component	Function
1	Camera	Captures images of waste objects for detection and sorting.
2	Waste input box	Holds incoming waste materials before detection and sorting.
3	Linear rail system	Enables horizontal movement of the receiving box to transport classified waste to the correct collection locations.
4	Stepper motor	Provides precise motion control to position the waste input box at the designated classification location.
5	Servo motor	Controls the opening and closing mechanism of the input box to release waste into the appropriate classification bin.
6	Classification bins	Store classified waste, including paper, plastic, metal, and other waste categories.

Table 2. Hardware and software configuration of the proposed waste sorting system

Component	Specification
Edge computing device	Raspberry Pi 5 (RP5), 8 GB RAM
Operating system	64-bit Raspberry Pi OS
Deep learning (DL) model	YOLOv8n
Camera	Full HD 1080p USB webcam
Input image size	640 × 640 pixels
Microcontroller	Arduino Uno
Servo motor	MG996R
Stepper motor	KH56QM2-801, 1.8°/step
Stepper motor driver	TB6600

Figure 7 illustrates the operational flowchart of the proposed waste detection and sorting system. Initially, a waste object is placed in the receiving box, after which the camera captures an image. The image data are transmitted to the RP5,

where the YOLOv8n model performs real-time object detection. If no waste object is detected, the system waits and continues acquiring subsequent frames for further detection. Once an object is detected, the RP5 sends the detection result to the Arduino Uno via serial communication. The Arduino Uno then generates control signals for the stepper motor and servo motor to operate the mechanical sorting mechanism. Based on the received detection result, the stepper motor moves the receiving box to the position corresponding to the designated bin. The servo motor then opens the bottom flap of the box to release the object into the bin and closes the flap after the operation is completed. Once the sorting process is finished, the stepper motor returns the receiving box to its initial position, and the system enters a ready state for the next detection and sorting cycle. If the operation continues, a new object is placed in the receiving box, and the process is repeated. Otherwise, the system stops when a stop command

is issued, or the power supply is disconnected.

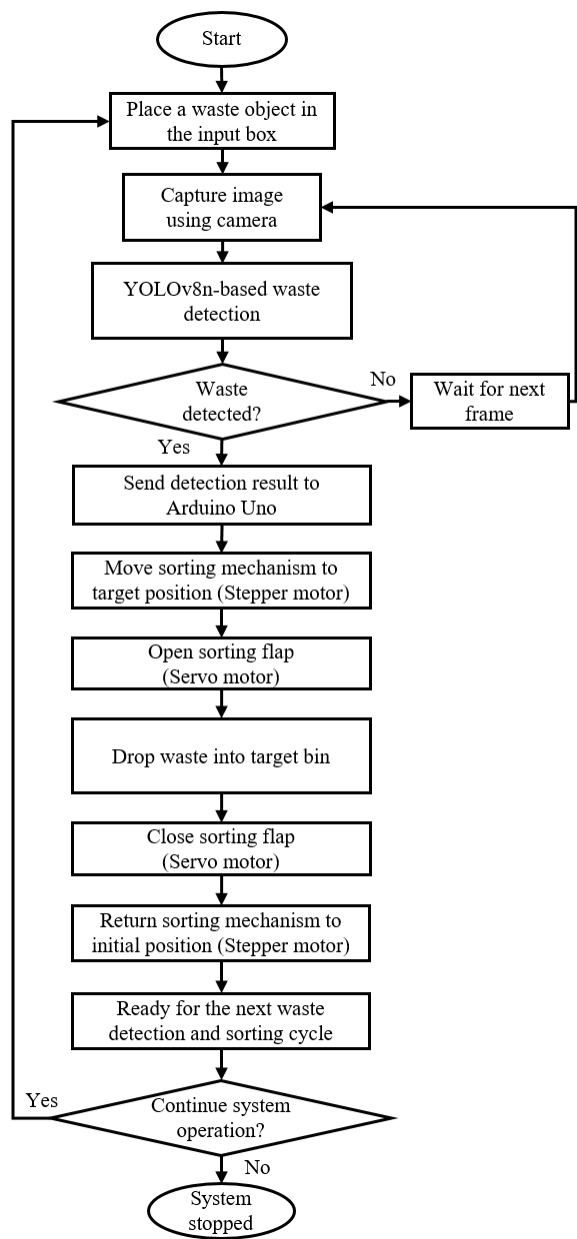


Figure 7. Operational flowchart of the waste detection and sorting system

3. RESULTS AND ANALYSIS

3.1 Dataset construction and model training

In this study, a custom dataset was constructed to support the training and evaluation of the proposed waste detection and sorting system. An overview of the dataset construction process is presented in Figure 8.

The waste data were collected using a USB webcam and a smartphone camera. All images were captured directly by the authors from real waste samples, without using images from any publicly available dataset. The initial dataset consisted of 800 original images, with 200 images for each waste category. The four categories considered were metal, paper, plastic, and other waste. Representative images of the four waste categories are shown in Figure 9. Specifically, class 0 (class_0), class 1 (class_1), class 2 (class_2), and class 3

(class_3) correspond to metal, paper, plastic, and other waste, respectively.

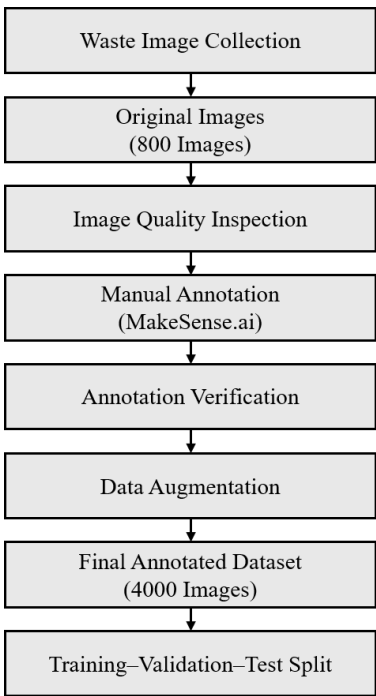


Figure 8. Dataset construction process

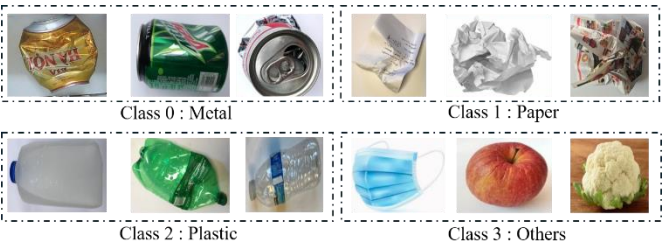


Figure 9. Sample images from the custom dataset representing four waste categories

To increase the diversity of the dataset, the images were collected under a range of conditions, including natural lighting, indoor lighting, different object orientations, varying camera-to-object distances, and diverse backgrounds. The waste samples were also placed at different positions and orientations within the image frame to reduce the likelihood that the model would learn features associated only with a fixed acquisition setup. Accordingly, brightness variations in the dataset were introduced directly during image acquisition through different lighting conditions, rather than by applying software-based brightness adjustments during data augmentation. After data collection, images that were blurred, underexposed, did not clearly show the target object, or had inadequate quality were inspected and removed. The remaining images were manually annotated using the MakeSense.ai online platform. Each waste object was assigned to its corresponding class and enclosed by a bounding box in the annotation format required by the YOLO model.

To ensure the quality of the dataset, all annotated images were manually reviewed. The review focused on the correctness of object class labels, the position and size of bounding boxes, and the identification of any missed objects. Cases involving incorrect class assignments, bounding boxes that did not adequately enclose the target objects, or

inconsistent annotations were corrected before the data were used for model training. This process helped improve the consistency and reliability of the dataset.

Because the number of original images was limited, the authors developed a Python-based data augmentation program in the PyCharm environment using the OpenCV library to expand the dataset. Unlike random data augmentation

methods, this study employed a deterministic augmentation strategy in which each original image was subjected to all four transformations: a 45° rotation, a 90° rotation, a horizontal flip, and a vertical flip, as illustrated in Figure 10. Each transformation was applied once to every original image. Therefore, the application probability of each transformation was 1.0, corresponding to 100%, as presented in Table 3.

Table 3. Data transformations used in this study

Transformation	Application Probability	Images per Original	Purpose
Retain the original image	100%	1	Preserve the original image information
45° rotation	100%	1	Increase diversity in object angles and orientations
90° rotation	100%	1	Increase diversity in object angles and orientations
Horizontal flip	100%	1	Generate additional horizontally symmetric viewing orientations
Vertical flip	100%	1	Generate additional vertically symmetric viewing orientations.
Total	-	5	One original image and four augmented images

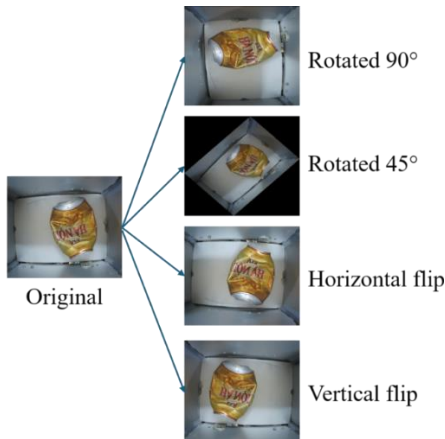


Figure 10. Illustration of the data augmentation techniques used to increase the diversity of the waste dataset

Variations in lighting conditions were introduced directly during image acquisition under natural and indoor lighting. Therefore, no software-based brightness adjustment was applied during the data augmentation stage. For each original image, four corresponding augmented images were generated. When combined with the original image, each source image produced a total of five images, increasing the number of images in each class from 200 to 1,000.

The above transformations were selected to introduce variations in object orientation and pose that may occur during practical sorting operations. Increasing data diversity helps reduce overfitting, improve model stability, and support better generalization within the evaluated conditions. After data augmentation, the final dataset comprised 4,000 images evenly distributed across the four classes, with 1,000 images per class, as presented in Table 4.

Table 4. Dataset statistics for the proposed waste detection and sorting system

Waste Category	Number of Images	Percentage (%)
Metal	1000	25%
Paper	1000	25%
Plastic	1000	25%
Others	1000	25%
Total	4000	100%

After quality inspection, annotation, and data augmentation, the final dataset comprised 4,000 images. The dataset was randomly divided into three subsets at a ratio of 80:10:10, including 3,200 images for the training set, 400 images for the validation set, and 400 images for the test set, as presented in Table 5. The training set was used to optimize the parameters of the YOLOv8n model. The validation set was used to monitor model performance during training and select the optimal model weights, whereas the test set was used to evaluate the final performance of the model on data that had not been used during training.

After completing data preprocessing, augmentation, and dataset splitting, the complete dataset was used to train the YOLOv8n model through the Ultralytics framework on the Google Colab platform. The training configuration and hyperparameters adopted in this study are summarized in Table 6.

Table 5. Dataset split for training, validation, and testing

Dataset	Number of Images	Percentage (%)
Training Set	3200	80%
Validation Set	400	10%
Test Set	400	10%
Total	4000	100%

Table 6. Training configuration and hyperparameters of the YOLOv8n model

Parameter	Value
Model	YOLOv8n
Number of Epochs	100
Image Size	640 × 640
Batch Size	16
Optimizer	SGD
Platform	Google Colab
GPU	NVIDIA Tesla T4
Framework	Ultralytics YOLOv8n

The training process was conducted for 100 epochs using input images resized to 640 × 640 pixels. A batch size of 16 was selected to balance computational efficiency and GPU memory utilization. Google Colab equipped with an NVIDIA Tesla T4 GPU was employed to accelerate the training process and reduce computation time. During training, the model learned representative visual features associated with different

waste categories, including paper, metal, plastic, and other waste. The best-performing model weights obtained during training were automatically saved and subsequently deployed on the RP5 platform to perform real-time waste detection and classification.

3.2 Evaluation metrics

In this study, several evaluation metrics were employed, including P, R, AP, mAP, F1, and FPS. These metrics were used to assess both the object detection performance and the real-time processing capability of the proposed system.

P measures the proportion of predicted objects that are correctly identified. It is expressed as [32]:

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

where, TP is the number of correctly detected objects, whereas FP is the number of detections that do not correspond to actual target objects.

R indicates the proportion of ground-truth objects successfully detected by the model. A higher R value implies that fewer target objects remain undetected. It is defined as follows [32]:

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

where, FN denotes the number of ground-truth objects that the model fails to detect.

AP summarizes the detection performance for an individual object class by computing the area under its Precision–Recall curve. AP can be expressed as [33]:

$$AP = \int_0^1 P(R)d(R) \quad (3)$$

mAP provides an overall assessment across all object classes by averaging their individual AP values [32]:

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i \quad (4)$$

where, N denotes the number of object classes and AP_i represents the Average Precision obtained for the i -th class.

In this study, detection performance was assessed using two mAP metrics: mAP@0.5 and mAP@0.5:0.95. The mAP@0.5 metric represents the mAP obtained at a single IoU threshold of 0.50. By contrast, mAP@0.5:0.95 is calculated by averaging the mAP values over ten IoU thresholds ranging from 0.50 to 0.95 in steps of 0.05. Consequently, mAP@0.5:0.95 provides a stricter and more comprehensive assessment of the model's detection and bounding-box localization performance under different overlap requirements. In object detection tasks, IoU quantifies the spatial agreement between a predicted bounding box (B_P) and its corresponding ground-truth bounding box (B_G). It is defined as the ratio of their intersection area to their union area [32]:

$$IoU = \frac{area(B_P \cap B_G)}{area(B_P \cup B_G)} \quad (5)$$

F1 is a metric that combines P and R through their harmonic mean, providing a balanced assessment of the detection model's performance. F1 is calculated as follows [33]:

$$F1 = \frac{2 \times P \times R}{P + R} \quad (6)$$

FPS indicates the number of image frames that the system can process per second. This metric is particularly important for evaluating the real-time performance of the proposed system when deployed on the RP5 embedded platform.

3.3 Model training and evaluation results

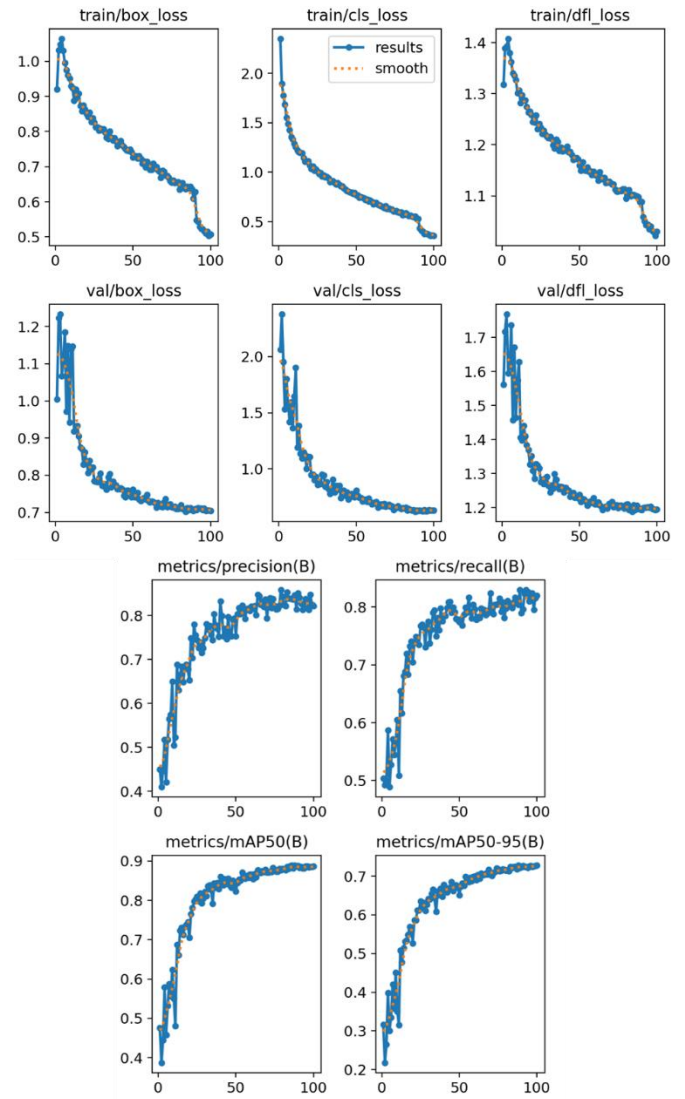


Figure 11. Training and validation curves of the YOLOv8n model, including loss functions, Precision, Recall, and mAP metrics

Figure 11 illustrates the training and validation results of the proposed YOLOv8n-based waste detection model throughout 100 training epochs. The results indicate that the training process was stable and that the model converged effectively. The training loss functions, including box loss (box_loss), classification loss (cls_loss), and DFL (dfi_loss), all exhibited a gradual decreasing trend as the number of epochs increased. Specifically, the box loss decreased from approximately 1.06 to 0.51, while the classification loss decreased significantly

from approximately 2.35 to below 0.4. Similarly, the dfl_loss continuously decreased, indicating improvements in both bounding box regression and object localization performance during training. The validation loss curves followed trends similar to those observed in the training set without significant fluctuations. This suggests that the model did not suffer from severe overfitting and maintained relatively good generalization capability on unseen data.

Regarding the detection performance of YOLOv8n, P, R, and mAP generally improved during training and became more stable after approximately 70-80 epochs. At epoch 100, both P and R reached approximately 0.82. However, the checkpoint corresponding to the epoch with the best performance on the validation set was selected for deployment. This checkpoint achieved a P of 0.847, a R of 0.805, an mAP@0.5 of 0.888, and an mAP@0.5:0.95 of 0.728. In addition, the F1-score obtained from the F1-confidence curve was 0.82.

These results indicate that YOLOv8n maintained a good balance among reducing false-positive predictions, detecting target objects, and localizing bounding boxes across multiple IoU thresholds. After approximately 70 epochs, improvements in the evaluation metrics gradually diminished, and the curves began to converge, indicating that the model was approaching a stable state under the current training configuration. Therefore, 100 epochs were considered sufficient to monitor convergence and select the best-performing checkpoint. To reduce computational time, future studies may use fewer training epochs or apply an early-stopping mechanism while maintaining stable detection performance.

During both training and inference, the YOLOv8n model utilizes internal class labels ranging from class_0 to class_3. In this study, class_0, class_1, class_2, and class_3 correspond to metal, paper, plastic, and other waste categories, respectively. Figure 12 presents the P-R curves of the YOLOv8n model for each waste category as well as the overall system performance. The P-R curve is a widely used evaluation tool in object detection tasks, illustrating the trade-off between Precision and Recall under different confidence thresholds. In Figure 12, the curve of class_0 is represented in light blue, class_1 in orange, class_2 in green, class_3 in red, while the overall curve for all classes is represented in dark blue. As shown in Figure 12, the proposed model achieved an overall mAP@0.5 value of 0.888, indicating effective waste detection capability. Most P-R curves are located near the upper-right region of the graph, reflecting high Precision and Recall performance across different waste categories.

Class-wise analysis reveals that the plastic category achieved the highest AP of 0.941, followed by paper with 0.930, metal with 0.885, and the other-waste category with 0.795. These results indicate that the proposed YOLOv8n model delivers stable and reliable detection performance across most waste categories. However, the other-waste category remains more challenging to detect due to its substantial diversity in shape, color, and visual appearance compared to the remaining classes. Moreover, the smooth and gradually decreasing shapes of the P-R curves demonstrate stable model behavior without abrupt performance degradation, maintaining a favorable balance between Precision and Recall.

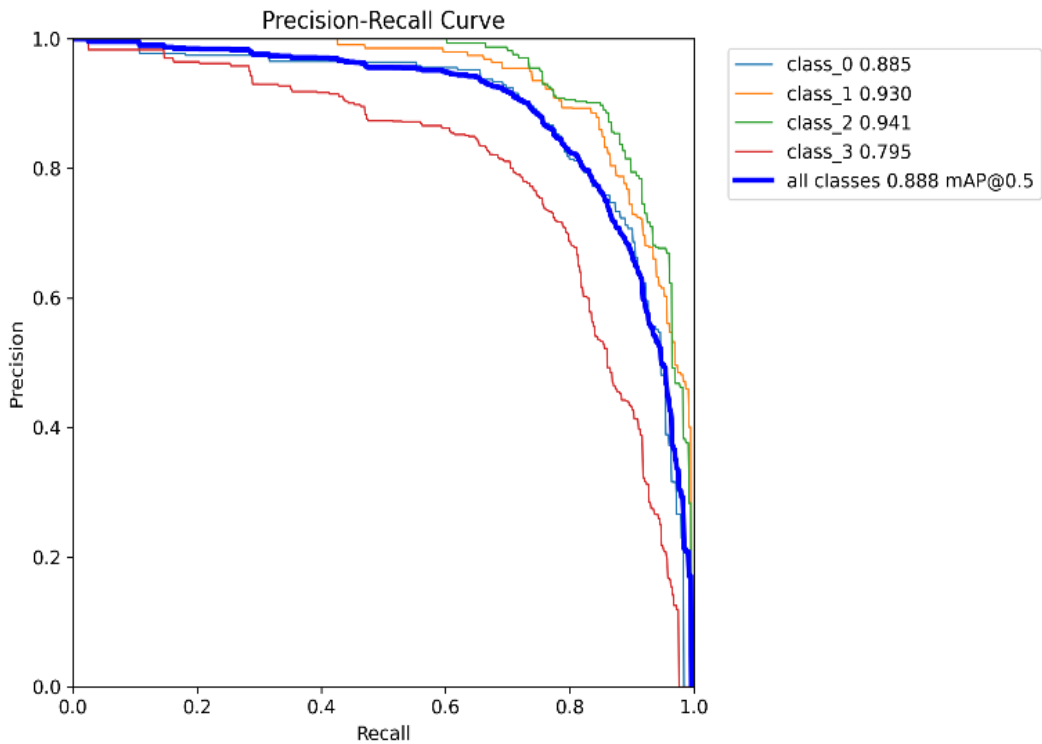


Figure 12. Precision-recall curves of the YOLOv8n-based waste detection model for all object classes

Figure 13 presents the Normalized Confusion Matrix of the YOLOv8n model on the test dataset. The results show that the model achieved relatively high correct classification rates across the object classes, with the values along the main diagonal ranging from 0.79 to 0.88. Specifically, the metal (class_0) and paper (class_1) classes achieved the highest

correct prediction rates, both at 0.88, followed by plastic (class_2) at 0.85, while other waste (class_3) achieved approximately 0.79. Overall, confusion among the object classes was relatively low, indicating that the model could effectively distinguish among different waste categories within the test dataset. However, some confusion remained

between the other-waste class and the background. Notably, among false-positive detections associated with the background, other waste and metal accounted for proportions of 0.51 and 0.23, respectively, indicating that these two classes contributed the majority of erroneous detections on the background. Conversely, other waste also exhibited the

highest false-negative rate, with approximately 12% of actual objects in this class remaining undetected and being classified as background. These errors may be associated with complex environmental conditions, such as light reflections, irregular object shapes, or visual similarities between the objects and the background.

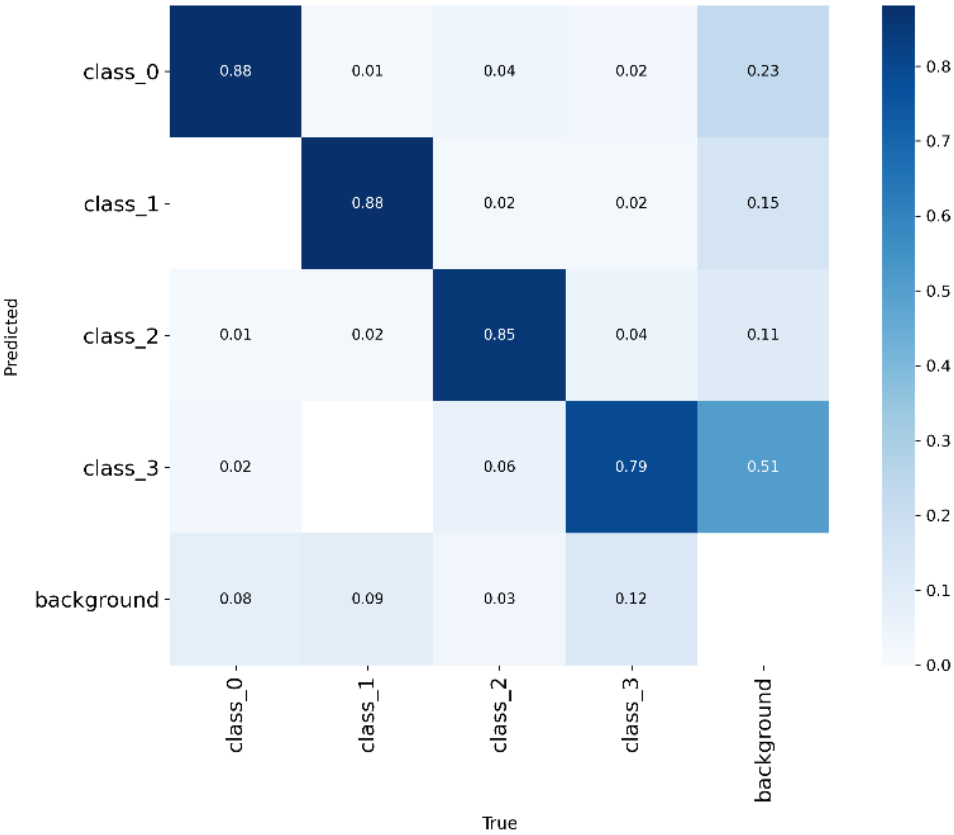


Figure 13. Normalized confusion matrix of the YOLOv8n model on the test dataset

The confusion matrix was analyzed in greater depth to identify potential causes of the model’s classification errors. The results show that the metal, paper, and plastic classes achieved relatively high correct classification rates of 0.88, 0.88, and 0.85, respectively, whereas the other-waste class achieved only 0.79. The lower performance of the other-waste class may be associated with its high intra-class diversity and the heterogeneity of the visual characteristics of the objects included in this category. Unlike the other three classes, which consist of object groups with relatively consistent characteristics, the other-waste class contains a wide variety of materials, including rubber, fabric, wood, foam, composite materials, and other types of household waste. These objects differ considerably in shape, size, color, and surface texture. Therefore, when the number of representative samples for each object type remains limited, the model encounters greater difficulty in learning common and stable features for the entire class.

In addition, some objects in the other-waste class exhibit visual characteristics similar to those of paper or plastic. The confusion matrix shows that approximately 4% of actual other-waste objects were incorrectly predicted as plastic, while approximately 6% of actual plastic objects were misclassified as other waste. This bidirectional confusion may result from

similarities in color, shape, or surface texture between certain objects in the two classes. In addition to inter-class confusion, approximately 12% of actual other-waste objects were not detected and were classified as background. This result suggests that some objects in the other-waste class may be small, partially occluded, have low contrast with the background, or lack sufficiently distinctive visual features for accurate detection. At the same time, the high proportion of false-positive detections for the other-waste class on the background indicates that certain environmental details may exhibit characteristics similar to those of objects in this class.

Overall, the lower performance of the other-waste class may be associated with its high intra-class diversity, the limited number of representative samples for each object type, and visual similarities to the background or the other classes. To address these limitations, future studies will increase the number and variety of samples in the other-waste class and collect additional data under more diverse conditions, including different backgrounds, illumination intensities, degrees of occlusion, object overlap, and viewing angles. In addition, data augmentation strategies and methods for improving the model’s feature representation capability will be investigated to enhance recognition performance for highly diverse objects.

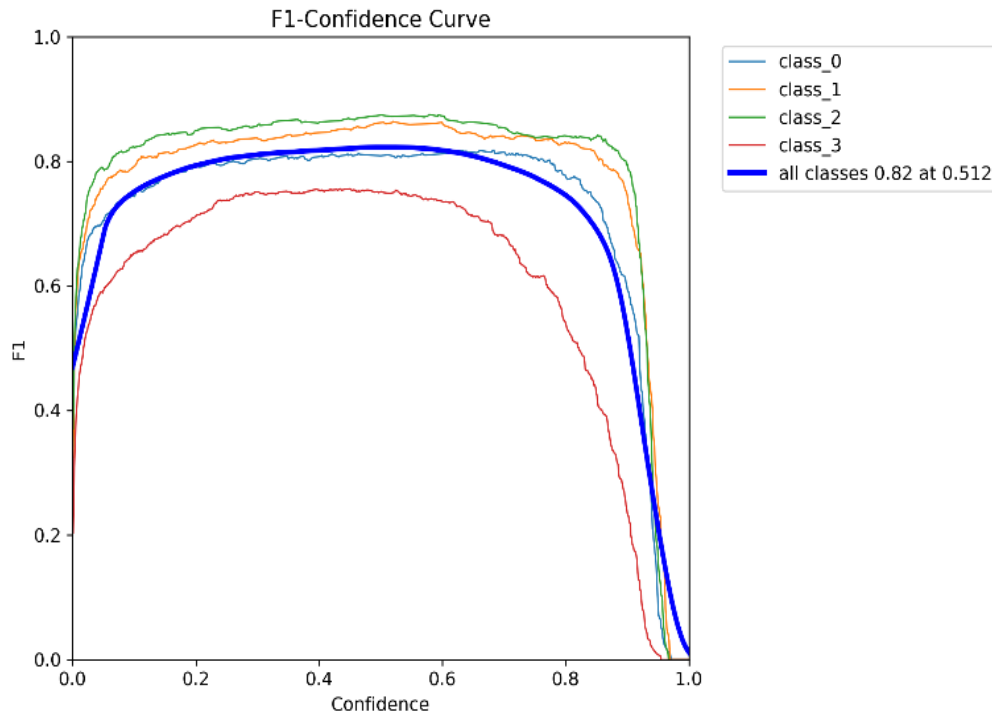


Figure 14. F1-confidence curves of the YOLOv8n model, showing class-wise performance and the optimal confidence threshold

Figure 14 illustrates the F1-confidence curves of the YOLOv8n model for each waste category as well as the overall system performance. The F1-score represents the harmonic mean of Precision and Recall, providing a balanced assessment of detection performance across different confidence thresholds. The results indicate that the overall F1-score reaches a maximum value of approximately 0.82 at a confidence threshold of around 0.512, suggesting that this threshold represents the optimal operating point of the model where Precision and Recall are effectively balanced. At lower confidence thresholds, the F1-score increases rapidly due to improved Recall as more objects are detected. However, if the threshold becomes too low, the number of false-positive predictions increases, leading to reduced Precision. Conversely, at higher confidence thresholds, the F1-score decreases as Recall declines because fewer detections are retained. Class-wise analysis shows that the plastic category (class_2) achieves the highest F1-score and maintains stable performance across a wide range of confidence thresholds, followed by the paper category (class_1) and the metal category (class_0). In contrast, the other-waste category (class_3) exhibits noticeably lower F1-scores, particularly at higher confidence thresholds, reflecting the greater difficulty associated with detecting this category due to its diversity in shape, color, and visual characteristics. Furthermore, the smooth and stable F1-confidence curves indicate reliable model behavior without sudden performance fluctuations. The relatively broad plateau region around the optimal threshold

also demonstrates that the model can maintain stable performance despite minor variations in the confidence threshold.

Overall, the offline evaluation results, including loss convergence, Precision–Recall curves, the confusion matrix, and F1-score analysis, demonstrate that the YOLOv8n model achieved satisfactory detection performance. With an optimal confidence threshold of approximately 0.512, the model effectively balances Precision and Recall, thereby improving correct object detection while reducing false detections. The quantitative results indicate that the trained YOLOv8n model possesses good generalization capability, stable detection performance, and high reliability. These findings provide a solid foundation for deploying the proposed real-time waste detection and classification system on the RP5 embedded platform.

Three lightweight object detection models, namely YOLOv5n, YOLOv7-tiny, and YOLOv8n, were evaluated under the same conditions. All three models were trained using the same dataset, an input image size of 640×640 pixels, 100 epochs, and the same main training settings. After training, the checkpoint corresponding to the epoch with the best performance on the validation set was selected for each architecture for evaluation and comparison. The evaluation metrics included P, R, mAP@0.5, mAP@0.5:0.95, and F1-score. The F1-score was obtained from the overall F1–confidence curve for all classes of each model at the optimal confidence threshold. The results are presented in Table 7.

Table 7. Performance comparison of lightweight object detection models

Model	P	R	mAP@0.5	mAP@0.5:0.95	F1-Score
YOLOv5n	0.832	0.819	0.888	0.725	0.82
YOLOv7-tiny	0.830	0.755	0.851	0.657	0.78
YOLOv8n	0.847	0.805	0.888	0.728	0.82

The comparison results in Table 7 show that YOLOv5n and YOLOv8n outperformed YOLOv7-tiny on most evaluation

metrics. Specifically, YOLOv8n achieved the highest Precision of 0.847, compared with 0.832 for YOLOv5n and

0.830 for YOLOv7-tiny. This result indicates that YOLOv8n was more effective at reducing false-positive predictions under the evaluated conditions, thereby improving the reliability of the detected objects. In terms of Recall, YOLOv5n achieved the highest value of 0.819, followed by YOLOv8n at 0.805 and YOLOv7-tiny at 0.755. The Recall difference between YOLOv5n and YOLOv8n was 0.014, equivalent to only approximately 1.4 percentage points, indicating that the detection capabilities of the two models were nearly comparable.

Regarding overall detection performance, YOLOv5n and YOLOv8n both achieved an mAP@0.5 of 0.888, exceeding the value of 0.851 obtained by YOLOv7-tiny. When evaluated across multiple IoU thresholds using mAP@0.5:0.95, YOLOv8n achieved the highest value of 0.728, slightly surpassing YOLOv5n at 0.725 and outperforming YOLOv7-tiny at 0.657. This result indicates that YOLOv8n maintained marginally better bounding-box localization quality across multiple IoU thresholds.

For the F1-score obtained from the F1-confidence curve, YOLOv5n and YOLOv8n both achieved a value of 0.82, indicating that the two models maintained a comparable balance between Precision and Recall at their respective optimal confidence thresholds. In contrast, YOLOv7-tiny achieved an F1-score of 0.78, which was lower than those of the other two models and consistent with its lower Recall value.

Overall, the results show that YOLOv8n achieved the highest Precision and mAP@0.5:0.95, while maintaining mAP@0.5 and F1-score values equivalent to those of YOLOv5n. Although its Recall was slightly lower than that of YOLOv5n, YOLOv8n still demonstrated a good balance among reducing false-positive predictions, detecting objects, and localizing bounding boxes across multiple IoU thresholds.

During training, the best YOLOv8n checkpoint was obtained at epoch 88, whereas the best YOLOv5n checkpoint was obtained at epoch 100. This result indicates that, under the current evaluation setting, YOLOv8n reached its best validation performance earlier than YOLOv5n. However, because both models were trained for the full 100 epochs, this finding only reflects the epoch at which the best checkpoint occurred and does not imply a reduction in actual training time. Based on the above analysis, YOLOv8n was selected for deployment in the proposed waste detection and sorting system because it achieved the highest Precision and mAP@0.5:0.95 while maintaining mAP@0.5 and F1-score values comparable to those of YOLOv5n.

3.4 Experimental results

Figure 15 presents the results of single waste-object detection on a computer using the YOLOv8n model under the same lighting conditions and viewing angle. The system was able to correctly detect waste categories such as metal, plastic, paper, and the other-waste group with high confidence. The results show that the model can localize the objects באמצעות bounding boxes and correctly identify the corresponding class labels. The experimental results are consistent with the quantitative evaluation metrics presented in Section 3.3, indicating that the model maintained stable detection performance across different waste categories. In the illustrated examples, the detection confidence reached approximately 0.94 for metal, 0.92 for plastic, 0.88 for paper, and 0.92 for the other-waste group. It can be observed that, for

heavily crumpled paper samples, the detection confidence tended to decrease slightly compared with some metal and plastic samples due to their irregular deformation and complex surface characteristics. Nevertheless, the model still successfully detected and correctly classified the investigated samples, demonstrating relatively stable performance under the established experimental conditions.

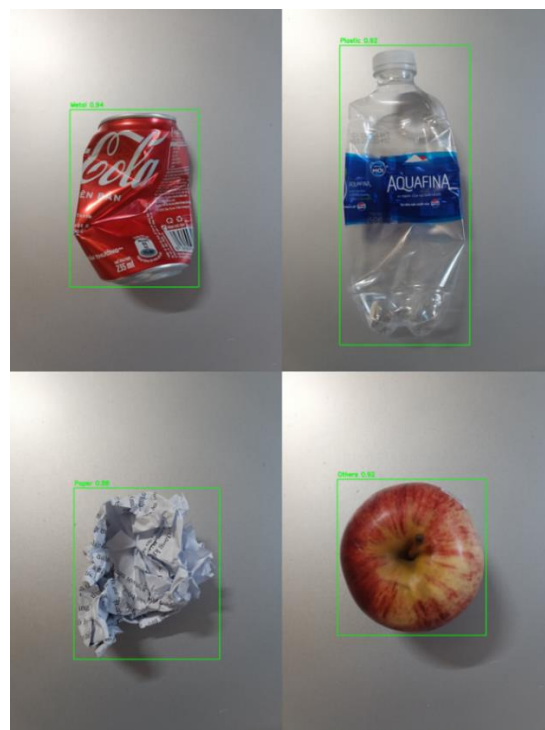


Figure 15. Single-object waste recognition results on a computer using the YOLOv8n model

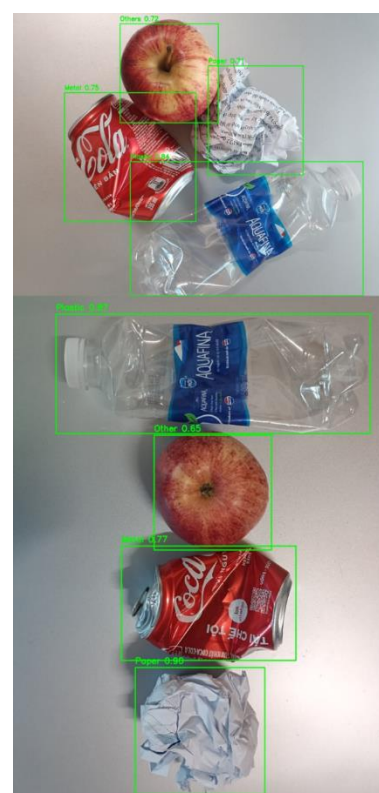


Figure 16. Mixed-waste recognition results on a computer using the YOLOv8n model

Figure 16 presents the results of detecting multiple waste objects appearing simultaneously within the same image frame using the YOLOv8n model. The experimental results show that the model can simultaneously detect different waste categories, including metal, plastic, paper, and other waste, even when the objects are positioned close together or partially overlap. The illustrated results demonstrate that the model can still accurately localize objects using bounding boxes and correctly identify the class labels of most objects with relatively stable confidence levels. Specifically, the detection confidence ranged from approximately 0.75 to 0.77 for metal, 0.84 to 0.87 for plastic samples, 0.71 to 0.90 for paper, and 0.65 to 0.72 for other waste. In cases where objects were partially occluded or overlapped, the detection confidence tended to decrease slightly because fewer visual features were available to the model. Nevertheless, the model maintained accurate classification for most objects in the images, thereby demonstrating its ability to effectively process multi-object scenes with more complex spatial arrangements than those containing only a single object. These results indicate that the YOLOv8n model has the potential to meet the requirements of the vision module in an intelligent waste detection and sorting system.

The above analyses indicate that the YOLOv8n model maintained stable waste detection performance in both single-object and multi-object scenarios. The model was still able to detect objects under partial occlusion or overlap, thereby providing evidence of its capability to operate in more complex situations than those involving only a single object in the image. These results provide a basis for the development of an intelligent waste detection and sorting system. Owing to its lightweight architecture, YOLOv8n is suitable for deployment in edge AI applications. Based on the obtained results, the model was subsequently deployed on an RP5 and integrated into the prototype automatic waste detection and sorting system.



Figure 17. Waste detection results on RP5 using the YOLOv8n model

Figure 17 presents the waste detection results obtained using the YOLOv8n model deployed directly on the RP5 hardware platform. The experimental results show that the system can correctly detect different waste categories, including metal, paper, plastic, and other waste, under practical operating conditions. The model can localize objects using bounding boxes and correctly classify them into the

corresponding classes with relatively stable confidence levels. Specifically, the system achieved confidence scores of approximately 0.82 for metal, 0.83 for paper, 0.80 for plastic, and 0.84 for other waste. It can be observed that the images in Figure 17 were acquired during different operating trials and therefore exhibit natural variations in illumination intensity, the distribution of bright and dark regions in the background, and the position, orientation, and shape of the objects. Compared with the experimental results obtained on the computer, the detection confidence on the RP5 tended to decrease slightly, possibly due mainly to practical operating factors such as lighting conditions, background noise, or mechanical vibrations. Nevertheless, the system correctly detected the investigated waste samples, demonstrating that the model can perform inference directly on the RP5 without relying on external computing resources.

Because the current prototype is designed to receive and process only one object at a time, multiple overlapping objects do not appear within the field of view of the sorting mechanism. However, the ability of the YOLOv8n model to handle more complex spatial arrangements was investigated in Figure 16, where multiple waste categories appeared within the same image frame, were distributed non-uniformly, and exhibited partial occlusion or overlap. In these cases, the model still correctly localized and classified most objects, although the prediction confidence tended to decrease when some visual features were obscured. The combined results in Figures 16 and 17 indicate that the YOLOv8n model maintained relatively stable detection performance despite variations in image acquisition conditions and object arrangements.

With an input image size of 640 × 640 pixels, the YOLOv8n model deployed on the RP5 achieved an inference speed of 6.69 ± 0.43 FPS, corresponding to approximately 6–7 FPS. In the proposed system, a waste object is first placed in the receiving box, where the camera captures its image and transmits it to the RP5 for inference and object detection using the YOLOv8n model. After inference is completed, the RP5 sends the detection result to the Arduino Uno via serial communication. The Arduino Uno then generates control signals for the stepper motor and servo motor to operate the mechanical sorting mechanism. Based on the detection result, the stepper motor moves the receiving box to the position corresponding to the designated bin, while the servo-actuated opening and closing mechanism releases the waste into the appropriate sorting compartment. Therefore, the system performs sorting sequentially through the detection and mechanical actuation stages rather than requiring continuous image processing at a very high frame rate. Accordingly, an inference speed of approximately 6–7 FPS is considered sufficient to ensure stable operation and satisfy the real-time waste sorting requirements of the proposed system.

Table 8. Computational complexity and resource utilization of YOLOv8n on the RP5

Metric	Value
Model size	6.0 MB
Parameters	3011628
FLOPs	8.2 GFLOPs
Average inference latency	150.47 ± 15.33 ms
Average frames per second (FPS)	6.69 ± 0.43
Average central processing unit (CPU) utilization	54.61 ± 4.34%
Average RAM usage	790.72 ± 1.68 MB

To quantitatively evaluate the feasibility of deploying the model on an edge device, the computational complexity and resource utilization of YOLOv8n on the RP5 were analyzed. The evaluated metrics included model size, number of parameters, FLOPs, inference latency, processing speed, CPU utilization, and RAM usage. The results are presented in Table 8.

YOLOv8n has a model size of approximately 6.0 MB, consists of 3,011,628 parameters, and requires approximately 8.2 GFLOPs for an input image size of 640×640 pixels. When deployed on an RP5 equipped with 8 GB of RAM, the model achieved an average inference latency of 150.47 ± 15.33 ms and an average processing speed of 6.69 ± 0.43 FPS, corresponding to approximately 6–7 FPS. The average CPU utilization and RAM usage were $54.61 \pm 4.34\%$ and 790.72 ± 1.68 MB, respectively. The resource utilization metrics were recorded directly during continuous inference on the RP5 using operating system monitoring tools, after unnecessary background processes had been disabled to minimize their influence on the measurements. The latency and FPS values reflect only the model inference process and do not include the time required for image acquisition, transmission of detection results, or operation of the mechanical mechanism. The results demonstrate that the computational and memory requirements

of YOLOv8n are within the capabilities of the RP5, while the model maintains detection performance suitable for the system's sequential sorting process. These experimental results confirm the feasibility of deploying YOLOv8n for a waste detection and sorting system on an embedded edge platform.

Figure 18 illustrates the experimental operation of the system through four sequential steps. In Step 1, the waste object, which in this case is an apple belonging to the other-waste class, is placed in the receiving box directly beneath the camera for image acquisition and real-time inference using the YOLOv8n model deployed on the RP5. Next, in Step 2, after the detection result is confirmed, the RP5 transmits the detection result to the Arduino Uno via serial communication. The Arduino Uno then controls the stepper motor to move the sorting mechanism along the horizontal guide rail to the correct position above the corresponding waste bin. In Step 3, the servo motor is activated to open the bottom flap of the box, allowing the waste object to fall into the designated bin and leaving the receiving box empty. Finally, in Step 4, the bottom flap is closed, and the stepper motor drives the sorting mechanism back to its initial position on the left, thereby completing a closed-loop operating cycle and preparing the system for the next detection and sorting operation.

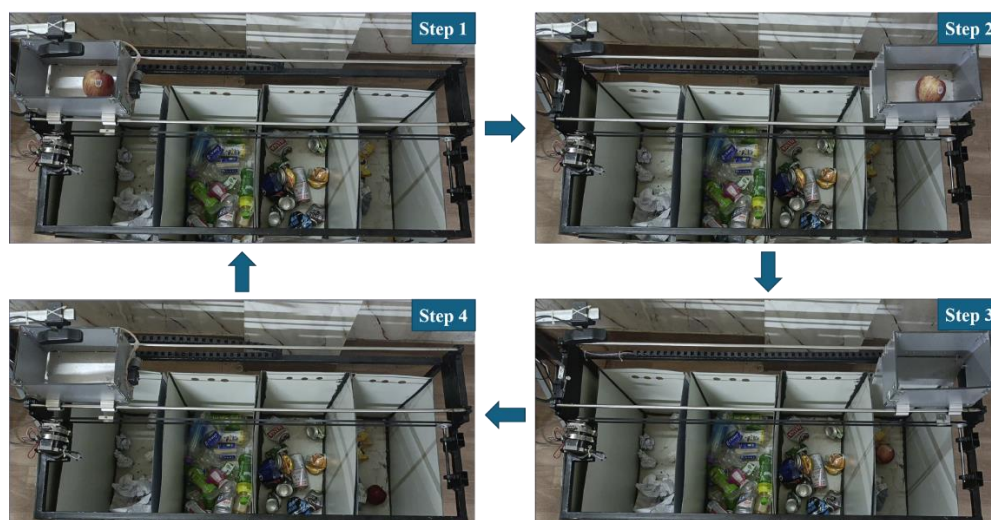


Figure 18. Illustration of the experimental operation process of the waste detection and sorting system

Following the operating procedure described above, Figure 19 presents the results of the physical sorting process for the four target waste categories. Each row illustrates two key states at the designated bin. The left column shows the moment when the sorting mechanism carrying the object stops at the correct position, while the right column shows the state immediately after the bottom flap opens to release the waste into the bin. The practical tests demonstrate that the sorting process was performed correctly. For metal waste in Row 1, the beverage can was accurately discharged into the third bin. For paper waste in Row 2, the paper sample was delivered to the first bin on the far left. For plastic waste in Row 3, the plastic bottle was successfully released into the second bin. For the other-waste category in Row 4, the apple was correctly deposited into the fourth bin. These visual results demonstrate that the YOLOv8n model, RP5, Arduino Uno, and actuation mechanism can operate in coordination to direct the investigated objects to their corresponding bins.

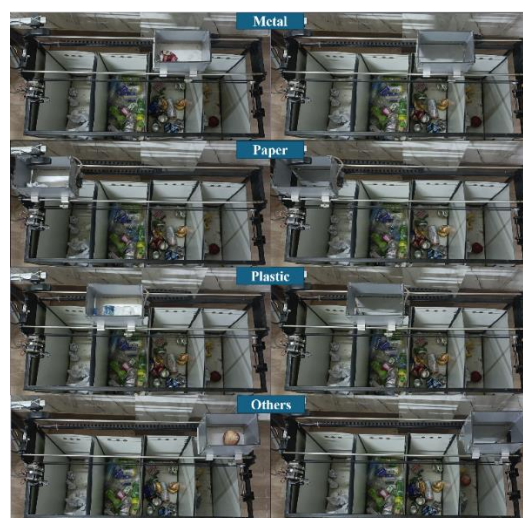


Figure 19. Experimental results of the waste detection and sorting system for the four target waste categories

Although inference speed reflects the model’s ability to process individual frames, this metric does not represent the number of complete sorting cycles that the system can perform per second. A complete operating cycle also includes the continuous acquisition and processing of image frames, confirmation of the detection result, transmission of the detection result, and operation of the actuators. Therefore, to provide a more comprehensive evaluation of the overall prototype performance, the execution time of each stage in a sorting cycle was measured separately. The evaluated stages included image processing and decision-making time, the time required for the stepper motor to move to the sorting position, servo motor operating time, and the time required for the stepper motor to return to its initial position. The results are summarized in Table 9.

As presented in Table 9, the vision processing and decision-making time ranged from 2.1 to 3.2 s, while the total time required to complete one sorting cycle ranged from 5.6 to 18.6 s. The vision processing and decision-making time includes

image acquisition, sequential frame processing, class-label confirmation, and transmission of the detection result to the control module. Therefore, this value is greater than the average single-frame inference latency of 150.47 ± 15.33 ms. The variation in total cycle time mainly depends on the position of each bin and the corresponding travel distance of the actuation mechanism. The paper category had the shortest cycle time because the stepper motor was not required to move to another position, whereas the other-waste category had the longest cycle time because the receiving box had to travel to the farthest position and then return to its initial position. Distinguishing between per-frame inference latency and the time required to complete an entire sorting cycle provides a more accurate description of the real-time characteristics of the proposed system. Because the system operates sequentially and processes each object individually, an inference speed of approximately 6–7 FPS is sufficient to provide the detection result to the control module before the actuation mechanism performs the sorting operation.

Table 9. Time analysis of the stages in a waste detection and sorting cycle

Waste Category	Stepper Motor Travel to Sorting Position (s)	Stepper Motor Return Time (s)	Servo Motor Operation Time (s)	Vision Processing and Decision Time (s)	Total Cycle Time (s)
Paper	0.0	0.0	3.5	2.1	5.6
Plastic	3.3	3.0	3.3	2.8	12.4
Metal	4.7	4.3	3.6	2.5	15.1
Other waste	6.5	5.4	3.5	3.2	18.6

In addition to the execution time of each cycle, the stability of the system during continuous operation was evaluated using the experimental prototype. A total of 150 objects from the four waste categories were introduced into the system in random order and processed consecutively without restarting the hardware or software throughout the experiment. During operation, the RP5 continuously acquired images and performed inference using the YOLOv8n model. Once the object class label was confirmed, the detection result was transmitted to the Arduino Uno via serial communication to control the actuation mechanism and direct the object into the corresponding bin. The results of the continuous-operation test are summarized in Table 10. In this experiment, a trial was considered successful when the object was assigned the correct class label by the model and delivered to the corresponding bin by the actuation mechanism without mechanical jamming.

Table 10. Results of the continuous-operation test of the waste detection and sorting system

Metric	Value
Number of test samples	150
Number of correctly sorted samples	137
Number of detection-label errors	9
Number of mechanical jams	4
Successful sorting rate	91.3%
Software failures	0
Communication failures	0
Hardware failures	0

As presented in Table 10, the system successfully completed 137 of 150 trials, corresponding to a successful sorting rate of 91.3%. Among the unsuccessful trials, nine cases were associated with incorrect class-label identification, while four involved temporary mechanical jams. Throughout

the experiment, no software failures, communication losses between the RP5 and Arduino Uno, or hardware failures were recorded. These results indicate that the system maintained relatively stable operation while processing objects consecutively under the established operating conditions.

To evaluate the power consumption of the proposed system, the main hardware components were divided into two independent load branches and measured separately during operation. The 5 VDC branch included the RP5, Full HD 1080p USB webcam, Arduino Uno, and MG996R servo motor. These devices were powered by a 220 VAC/5 VDC power adapter. A WITRN K2 USB power meter was connected in series with the 5 VDC supply line of the entire branch; therefore, the measured value represents the total power consumption of these components.

The 24 VDC branch included the TB6600 driver and KH56QM2-801 stepper motor. This branch was powered by a 220 VAC/24 VDC power adapter. A KEWEISI KWS-DC200 DC power meter was connected in series between the output of the 24 VDC power supply and the input of the TB6600 driver. The recorded value represents the combined power consumption of the TB6600 driver and the stepper motor during the sorting motion.

Figure 20 illustrates the connection diagram and arrangement of the power measurement devices, while the measured power consumption of the two functional branches is summarized in Table 11.

The processing and control unit, including the RP5, USB camera, Arduino Uno, and servo motor, consumed approximately 7.57 W. Meanwhile, the stepper motor and TB6600 driver consumed approximately 11.57 W. The total power consumption of the entire system under the investigated operating conditions was approximately 19.14 W. These results show that the prototype can operate at a power level below 20 W, further supporting the feasibility of deploying the waste sorting system on an edge computing platform.

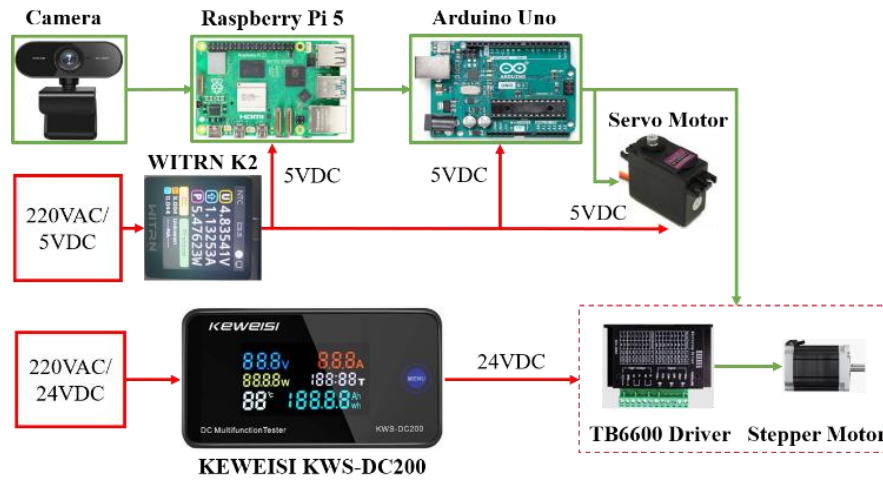


Figure 20. Arrangement of the power measurement devices

Table 11. Power consumption of the proposed waste detection and sorting system

Component	Voltage	Current	Power
Raspberry Pi 5 (RP5) + camera + Arduino Uno + servo motor	$\approx 5.14 \text{ V}$	$\approx 1.47 \text{ A}$	$\approx 7.57 \text{ W}$
Stepper motor + TB6600 driver	$\approx 22.5 \text{ V}$	$\approx 0.51 \text{ A}$	$\approx 11.57 \text{ W}$
Total system			$\approx 19.14 \text{ W}$

In summary, the results regarding detection performance, inference speed, cycle time, continuous-operation capability, and power consumption demonstrate that the YOLOv8n model, RP5, Arduino Uno, and actuation mechanism can operate in coordination to sequentially perform waste detection and sorting. The system achieved a successful sorting rate of 91.3% over 150 consecutive trials, with an inference speed of approximately 6–7 FPS and a total power consumption of approximately 19.14 W. Although several detection-label errors and temporary mechanical jams occurred, no software failures, communication losses, or hardware failures were recorded during the experiment. The obtained results demonstrate the feasibility of integrating edge AI with a mechatronic system in an intelligent waste detection and sorting prototype. In future studies, the number and diversity of test samples will be expanded, the duration of continuous operation will be extended, and the reliability of both the vision module and the mechanical mechanism will be further improved.

4. CONCLUSIONS AND FUTURE WORK

This study designed and implemented an integrated waste detection and mechanical sorting system using the YOLOv8n model. On a dataset comprising 4,000 images from four categories-paper, metal, plastic, and other waste-the best YOLOv8n checkpoint achieved a Precision of 0.847, a Recall of 0.805, an mAP@0.5 of 0.888, an mAP@0.5:0.95 of 0.728, and an F1-score of 0.82. When deployed on a RP5 using CPU-only inference, the model achieved a processing speed of $6.69 \pm 0.43 \text{ FPS}$, an inference latency of $150.47 \pm 15.33 \text{ ms}$, CPU utilization of $54.61 \pm 4.34\%$, and RAM usage of $790.72 \pm 1.68 \text{ MB}$. During 150 consecutive operating trials, the system successfully completed 137 trials, corresponding to a success rate of 91.3%, with a power consumption of approximately 19.14 W. These results demonstrate the feasibility of deploying a lightweight DL model on an edge computing

platform for waste detection and mechanical sorting. However, the study still has several limitations. Detection confidence tended to decrease for objects with highly variable and irregular shapes, such as heavily crumpled paper, as well as under complex variations in illumination and backgrounds containing multiple sources of visual interference. In addition, the current prototype processes only one object per cycle; the total cycle time remains dependent on the position of the target bin, and several temporary mechanical jams were observed. In future studies, the dataset will be expanded in terms of the number of samples, waste categories, and diversity of acquisition conditions, including illumination, background, viewing angle, object deformation, and occlusion. Techniques such as network pruning and hardware acceleration will be investigated to improve inference speed and reduce latency on edge devices. The mechanical design will also be improved by incorporating a waste feeding mechanism, supporting the consecutive processing of multiple objects, and integrating jam detection and recovery mechanisms to enhance continuous-operation capability and sorting throughput. The system should also be evaluated over a larger number of trials and for longer operating periods to assess its durability, stability, and deployment potential in practical industrial environments.

ACKNOWLEDGMENT

This research was funded by Hung Vuong University under grant number HV34.2024.

REFERENCES

- [1] Belyamani, I. (2025). Artificial intelligence in waste management systems: Applications, challenges, and prospects. *Waste Management Bulletin*, 3: 100269. <https://doi.org/10.1016/j.wmb.2025.100269>

- [2] Single, S., Iranmanesh, S., Raad, R. (2023). RealWaste: A novel real-life data set for landfill waste classification using deep learning. *Information*, 14(12): 633. <https://doi.org/10.3390/info14120633>
- [3] LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. *Nature*, 521(7553): 436-444. <https://doi.org/10.1038/nature14539>
- [4] Nayfeh, A., Al-Azani, S., Samma, H. (2025). A two-stage YOLOv8 approach for waste detection and classification in cognitive cities. *Transportation Research Procedia*, 86: 579-586. <https://doi.org/10.1016/j.trpro.2025.03.111>
- [5] Guan, Z.Y. (2023). Real time object recognition based on YOLO model. *Theoretical and Natural Science*, 28(1): 137-143. <https://doi.org/10.54254/2753-8818/28/20230450>
- [6] Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016). You only look once: Unified, real-time object detection. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, pp. 779-788. <https://doi.org/10.1109/CVPR.2016.91>
- [7] Rashida, J., Hamzah, R., Abu Samah, K.A.F., Ibrahim, S. (2022). Implementation of faster region-based convolutional neural network for waste type classification. In 2022 International Conference on Computer and Drone Applications (ICONDA), Kuching, Malaysia, pp. 125-130. <https://doi.org/10.1109/ICONDA56696.2022.10000369>
- [8] Arishi, A. (2025). Real-time household waste detection and classification for sustainable recycling: A deep learning approach. *Sustainability*, 17(5): 1902. <https://doi.org/10.3390/su17051902>
- [9] Di, J.X., Xi, K.K., Yang, Y. (2025). An enhanced YOLOv8 model for accurate detection of solid floating waste. *Scientific Reports*, 15: 25015. <https://doi.org/10.1038/s41598-025-10163-2>
- [10] Ren, Y.M., Li, Y.Z., Gao, X.Y. (2024). An MRS-YOLO model for high-precision waste detection and classification. *Sensors*, 24(13): 4339. <https://doi.org/10.3390/s24134339>
- [11] Castro-Bello, M., Roman-Padilla, D.B., Morales-Morales, C., et al. (2025). Convolutional neural network models in municipal solid waste classification: Towards sustainable management. *Sustainability*, 17(8): 3523. <https://doi.org/10.3390/su17083523>
- [12] Okano, M.T., Lopes, W.A.C., Ruggero, S.M., Vendrametto, O., Fernandes, J.C.L. (2025). Edge AI for industrial visual inspection: YOLOv8-based visual conformity detection using Raspberry Pi. *Algorithms*, 18(8): 510. <https://doi.org/10.3390/a18080510>
- [13] Ong, E.I.C., Flores, K.M.A., Pamittan, M.A.L., Rulona, M.F.M., Motin, C.N., Rosales, M.A. (2025). Robo-sort: Real-time object-based detection for automated solid waste collection and segregation using YOLOv8. In 2025 6th International Conference on Big Data Analytics and Practices (IBDAP), Chiang Mai, Thailand, pp. 14-18. <https://doi.org/10.1109/IBDAP65587.2025.11145869>
- [14] Li, P., Xu, J.Y., Liu, S.B. (2024). Solid waste detection using enhanced YOLOv8 lightweight convolutional neural networks. *Mathematics*, 12(14): 2185. <https://doi.org/10.3390/math12142185>
- [15] Liu, S.L., Chen, R.H., Ye, M.H., Luo, J.W., Yang, D.R., Dai, M. (2024). EcoDetect-YOLO: A lightweight, high-generalization methodology for real-time detection of domestic waste exposure in intricate environmental landscapes. *Sensors*, 24(14): 4666. <https://doi.org/10.3390/s24144666>
- [16] Dipo, M.H., Farid, F.A., Mahmud, M.S.A., et al. (2025). Real-time waste detection and classification using YOLOv12-based deep learning model. *Digital*, 5: 19. <https://doi.org/10.3390/digital5020019>
- [17] Rahmatulloh, A., Darmawan, I., Aldya, A.P., Nursuwars, F.M.S. (2025). WasteInNet: Deep learning model for real-time identification of various types of waste. *Cleaner Waste Systems*, 10: 100198. <https://doi.org/10.1016/j.clwas.2024.100198>
- [18] Partosan, K.N.L., Villanueva, E.J.R., Garcia, R.G. (2026). Campus-scale real-time waste classification on Raspberry Pi 5 using You Only Look Once version 8. *Engineering Proceedings*, 134: 28. <https://doi.org/10.3390/engproc2026134028>
- [19] Siyad, R.K., Islam, M.M., Nasseef, A.O., Ahmed, K.S., Hasan, M.S., Mashfy, M.M. (2026). AI-driven waste management system using deep learning model for detection and novel dual-axis actuator-based waste categorization. *Results in Engineering*, 30: 111286. <https://doi.org/10.1016/j.rineng.2026.111286>
- [20] Echeverry, A.P., López, C.F. (2024). Autorecycler: Prototype based on artificial vision to automate the material classification process (plastic, glass, cardboard and metal). *Mendeley Data*, V2. <https://doi.org/10.17632/yf8z2263gy.2>
- [21] Chacón-Albero, O., Campos-Mocholí, M., Marco-Detchart, C., Julian, V., Rincon, J.A., Botti, V. (2025). AI for sustainable recycling: Efficient model optimization for waste classification systems. *Sensors*, 25(12): 3807. <https://doi.org/10.3390/s25123807>
- [22] Ikram, S.T., Mohanraj, V., Ramachandran, S., Balakrishnan, A. (2023). An intelligent waste management application using IoT and a genetic algorithm-fuzzy inference system. *Applied Sciences*, 13(6): 3943. <https://doi.org/10.3390/app13063943>
- [23] Dinh, X.M., Luong, T.G., Le, X.H., Nguyen, V.T., Kim, D.T. (2025). Adaptive fuzzy dynamic surface control with sliding mode control for enhanced trajectory tracking of delta robots. *Journal of Control, Automation and Electrical Systems*, 36(4): 611-635. <https://doi.org/10.1007/s40313-025-01180-7>
- [24] Jocher, G., Qiu, J., Liu, M.Y., Lyu, S., Akyon, F.C., Kalfaoglu, M.E. (2026). Ultralytics YOLO26: Unified real-time end-to-end vision models. *arXiv preprint arXiv:2606.03748*. <https://doi.org/10.48550/arXiv.2606.03748>
- [25] Wang, G., Chen, Y.F., An, P., Hong, H.Y., Hu, J.H., Huang, T.G. (2023). UAV-YOLOv8: A small-object-detection model based on improved YOLOv8 for UAV aerial photography scenarios. *Sensors*, 23(16): 7190. <https://doi.org/10.3390/s23167190>
- [26] Zheng, Z.H., Wang, P., Liu, W., Li, J.Z., Ye, R.G., Ren, D.W. (2020). Distance-IoU loss: Faster and better learning for bounding box regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(7): 12993-13000. <https://doi.org/10.1609/aaai.v34i07.6999>
- [27] Li, X., Wang, W.H., Wu, L.J., et al. (2020). Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *arXiv preprint arXiv:2006.04388v1*. <https://doi.org/10.48550/arXiv.2006.04388>

- [28] Duc, Q.A.N., Kim, T.D., Nguyen, Q.C., et al. (2024). Optimizing traffic light control using YOLOv8 for real-time vehicle detection and traffic density. In 2024 9th International Conference on Integrated Circuits, Design, and Verification (ICDV), Hanoi, Vietnam, pp. 119-124. <https://doi.org/10.1109/ICDV61346.2024.10616901>
- [29] Dao, T.M.P., Vu, S.D., Nguyen, P.D.T., et al. (2026). Simulation-based control and vision prototype for a 5-DOF waste-sorting robot. *Journal Européen des Systèmes Automatisés*, 59(3): 759-774. <https://doi.org/10.18280/jesa.590319>
- [30] Ju, R.Y., Cai, W.M. (2023). Fracture detection in pediatric wrist trauma X-ray images using YOLOv8 algorithm. *Scientific Reports*, 13(1): 20077. <https://doi.org/10.1038/s41598-023-47460-7>
- [31] Gupta, C., Gill, N.S., Gulia, P., et al. (2025). An enhanced deep learning-based framework for diagnosing apple leaf diseases. *Scientific Reports*, 15(1): 39699. <https://doi.org/10.1038/s41598-025-23272-9>
- [32] Le, H.B., Kim, T.D., Ha, M.H., Tran, A.L.Q., Nguyen, D.T., Dinh, X.M. (2023). Robust surgical tool detection in laparoscopic surgery using YOLOv8 model. In International Conference on System Science and Engineering (ICSSE), Ho Chi Minh, Vietnam, pp. 537-542. <https://doi.org/10.1109/ICSSE58758.2023.10227217>
- [33] Yang, Y.Q., Pi, D.N., Wang, L.Y., et al. (2024). Based on improved YOLOv8 and Bot SORT surveillance video traffic statistics. *Research Square Preprint*. <https://doi.org/10.21203/rs.3.rs-4161504/v1>