sustainability

MDPI

Article

Convolutional Neural Network Models in Municipal Solid Waste Classification: Towards Sustainable Management

Mirna Castro-Bello [1], Dominic Brian Roman-Padilla [1], Cornelio Morales-Morales [2,*], Wilfrido Campos-Francisco [1], Carlos Virgilio Marmolejo-Vega [1], Carlos Marmolejo-Duarte [3], Yanet Evangelista-Alcocer [1] and Diego Esteban Gutiérrez-Valencia [1]

[1] Technological Institute of Chilpancingo, National Institute of Technology of Mexico, Chilpancingo de los Bravo 39090, Mexico; mirna.cb@chilpancingo.tecnm.mx (M.C.-B.); mg23520624@chilpancingo.tecnm.mx (D.B.R.-P.)
[2] Technological Institute of San Juan del Río, National Institute of Technology of Mexico, San Juan del Rio Queretaro 76800, Mexico
[3] Center of Land Policy and Valuations, Barcelona School of Architecture (ETSAB), Polytechnic University of Catalonia, 08034 Barcelona, Spain; carlos.marmolejo@upc.edu
* Correspondence: cornelio.mm@sjuanrio.tecnm.mx

Abstract: Municipal Solid Waste (MSW) management presents a significant challenge for traditional separation practices, due to a considerable increase in quantity, diversity, complexity of types of solid waste, and a high demand for accuracy in classification. Image recognition and classification of waste using computer vision techniques allow for optimizing administration and collection processes with high precision, achieving intelligent management in separation and final disposal, mitigating environmental impact, and contributing to sustainable development objectives. This research consisted of evaluating and comparing the effectiveness of four Convolutional Neural Network models for MSW detection, using a Raspberry Pi 4 Model B. To this end, the models YOLOv4-tiny, YOLOv7-tiny, YOLOv8-nano, and YOLOv9-tiny were trained, and their performance was compared in terms of precision, inference speed, and resource usage in an embedded system with a custom dataset of 1883 organic and inorganic waste images, labeled with Roboflow by delimiting the area of interest for each object. Image preprocessing was applied, with resizing to 640 × 640 pixels and contrast auto-adjustments. Training considered 85% of images and testing considered 15%. Each training stage was conducted over 100 epochs, adjusting configuration parameters such as learning rate, weight decay, image rotation, and mosaics. The precision results obtained were as follows: YOLOv4-tiny, 91.71%; YOLOv7-tiny, 91.34%; YOLOv8-nano, 93%; and YOLOv9-tiny, 92%. Each model was applied in an embedded system with an HQ camera, achieving an average of 86% CPU usage and an inference time of 1900 ms. This suggests that the models are feasible for application in an intelligent container for classifying organic and inorganic waste, ensuring effective management and promoting a culture of environmental care in society.

Keywords: object detection; waste; Convolutional Neural Networks; Raspberry Pi; embedded system; sustainable development; management

Academic Editor: Jian Tang

Received: 21 February 2025
Revised: 5 April 2025
Accepted: 8 April 2025
Published: 14 April 2025

Citation: Castro-Bello, M.; Roman-Padilla, D.B.; Morales-Morales, C.; Campos-Francisco, W.; Marmolejo-Vega, C.V.; Marmolejo-Duarte, C.; Evangelista-Alcocer, Y.; Gutiérrez-Valencia, D.E. Convolutional Neural Network Models in Municipal Solid Waste Classification: Towards Sustainable Management. *Sustainability* **2025**, *17*, 3523. https://doi.org/10.3390/su17083523

1. Introduction

The management of Municipal Solid Waste (MSW) is a major challenge, due to its high generation rate. In 2020, the total amount generated worldwide was estimated at 2.24 billion tons. This is attributed to rapid population growth and the ongoing advancement of

urbanization. Projections for the year 2050 indicate a 73% increase compared to the figures reported in 2020, reaching an estimated total of 3.88 billion tons [1]. The recognition and classification of Municipal Solid Waste (MSW) images offers the opportunity to identify and catalog them using computer vision techniques, optimizing intelligent and effective management processes to maintain urban environment quality, improve citizens' quality of life, and contribute to achieving sustainable development objectives [2]. On the other hand, modern object identification technology helps to improve object detection; for example, lightweight deep learning algorithms with reduced size achieve similar accuracy to models that require higher costs and requirements. These have achieved notable advances, standing out for their compact size and higher speed without a decrease in accuracy, and providing theoretical and technical support; they are particularly useful in integrated devices, allowing the achievement of superior levels of classification efficiency [3–5].

Convolutional Neural Networks (CNN) are deep learning models focused on image processing and hierarchical feature extraction, as well as spatial locality modeling. They consist of multiple convolutional and fully connected layers, facilitating the extraction of more detailed features and better image interpretation, and capturing, in turn, more global context information. By understanding the general structure and semantics of each model, its utility can be determined. Some examples of deep network structures are ResNet, Inception, EfficientNet, ShuffleNet, and YOLO [2]. ResNet-50 is a deep CNN architecture focused on solving problems such as performance degradation, learning difficulty, and convergence problems with increased layers. Applying residual connections to learn from mappings and extract detailed features from images, it consists of convolutional layers, batch normalization, pooling, and fully connected layers [6]. ShuffleNet v2 focuses on designing more efficient networks by optimizing memory usage and network speed, maintaining the same number of input and output channels of the convolutional layer to minimize memory access cost, reducing the use of group convolutions to save resources, simplifying the network branch structure to improve parallel operation, and decreasing element-level operations to improve network operation [7]. You Only Look Once (YOLO) is an object detection model designed for real-time image processing. By analyzing an image once while moving, it can detect and classify objects through prior training based on a labeled and cataloged dataset. It outperforms traditional CNNs in terms of efficiency for real-time object detection, by using a single CNN to predict bounding boxes and class probabilities simultaneously. In addition to having fast frame detection, it reasons globally, and performs rapid learning to identify and generalize object characteristics [8]. According to Ultralytics, YOLO approaches object detection as a single regression problem by predicting bounding boxes and class probabilities from full images in one evaluation. This approach makes YOLO faster than previous two-stage detectors, while still maintaining high precision, and it allows for rapid export, implementation, and execution on Raspberry Pi devices. YOLOv4-tiny, considered a lightweight model designed based on YOLOv4, uses less memory and operates faster, significantly increasing its viability for implementation in smart devices [9]. YOLOv7-tiny, the lightweight version of YOLOv7, stands out for its speed and efficiency, with 6.2 M parameters and 13.8 B operations (FLOPs). It is especially effective in detecting objects of various sizes, achieving performance scores of 18.8% for small objects, 42.4% for medium objects, and 51.9% for large objects [10]. YOLOv8n (nano) is a lightweight model designed for speed and efficiency, and is particularly suitable for devices with limited computing resources. It considers only 3.2 M parameters and 8.7 B FLOPs, is optimized to run on a CPU, and its smaller architecture makes it ideal for edge device implementations, while still maintaining good object detection performance [10]. Finally, YOLOv9t (tiny) is a simplified version of YOLOv9 designed for resource-limited applications, featuring 2.0 M parameters and 7.7 B FLOPs, with an inference time of 2.3 ms

on a T4 GPU with TensorRT10. This version is particularly useful for deployment on edge devices and mobile applications where computational resources are limited, without significantly sacrificing precision [10].

Literature Review

In the field of object detection models, notable advances have been reported in the literature. For example, the authors of [11] designed a model called GC-YOLOv5 for classifying batteries, orange peels, used paper, cups, and paper bottles. They used 642 images for training and 40 for validation, achieving an accuracy of 99.86% after 500 epochs, implementing their application with a JavaScript interface. Similarly, the authors of [12] developed the PublicGarbageNet algorithm with a final accuracy of 96.35%, which classifies four categories: kitchen waste, recyclables, hazardous waste, and other types. It is focused on multitasking and is based on CNN architecture, and was trained with a public dataset of 10,624 images and 10 classes (kitchen waste, plastic, paper, electronics, metal, glass, non-recyclable paper and plastics, textiles, and hazardous waste). Its performance was improved by optimizing the backbone, data augmentation, hyperparameter tuning, and label smoothing. Meanwhile, the authors of [13] proposed a waste classification algorithm that is resistant to the low accuracy caused by light and shadow interference, with adaptive capability in image lighting, using a threshold replacement method to reduce shadow noise, along with the Canny operator to help crop the white background in images. The algorithm was optimized based on the MLH-CNN model, achieving an accuracy of 96.77% with the authors' constructed dataset and 93.72% with TrashNet. References [14–16] developed a lightweight CNN model with an efficient algorithm on an embedded device for detecting surface defects in industrial products, reaching accuracy levels comparable to those of state-of-the-art (SOTA) models, while using fewer parameters and requiring fewer computations. Additionally, the authors implemented safe and optimal fault-tolerant control, based on neural networks, to reconstruct system dynamics from data, designing an adaptive critical scheme with a non-quadratic cost function to handle input constraints and reduce computational load.

With regard to datasets, the authors of [2] developed an integrated method of waste image recognition and classification that combines ResNet-50, YOLOv5, and weakly supervised CNN algorithms, improving the accuracy and efficiency of image recognition. ResNet-50 was used to extract features from the images, and weakly supervised CNNs were used for training and prediction. The authors evaluated four public datasets: HGI-30, TrashNet, GINI, and Public GarbageNet. With HGI-30, the inference time, FLOPs, and MAPE were reduced by more than 48.6%, 46.5%, and 41%, respectively, and an accuracy of 97.02% was achieved. Meanwhile, the authors of [17] presented a critical analysis of 11 existing waste datasets, collecting and summarizing previous studies by each author and their experimental results. Additionally, they designed two benchmark datasets, combining detect-waste, classify-waste, and all the categories identified in the study. Finally, they presented a two-stage detector for waste classification and localization, using EfficientDet-D2 to locate waste and EfficientNet-B2 to classify it through semi-supervised training using unlabeled images, achieving 70% average precision in waste localization and 75% in classification.

Another factor to consider is the comparative performance of models that can be applied in embedded systems. For example, the authors of [18] presented a model to classify inorganic containers in waste processing sites. They compared five CNN architectures: Xception, Inception V3, ResNet-50, ResNet-50 V2, and DenseNet-201, previously trained with a dataset of HDPE plastic, PET, glass bottles and jars, cans, cardboard, and flexible plastic containers, collected from three different Kaggle resources: a water bottle image

dataset, waste classification, and waste classification into 12 classes. For training, they considered data augmentation, rescaling, rotation, displacement, tilting, zoom, and flipping, obtaining a CNN model based on the DenseNet-201 architecture with an accuracy of 0.96%. Under this same approach, the contributions of [6] focused on creating a deep learning-based waste classification system that enables recognition and recycling of household waste through a lightweight garbage classification model, GCNet (Garbage Classification Network). The ShuffleNet v2 model was improved by integrating a parallel mixed attention mechanism (PMAM), new activation functions, and transfer learning, achieving an average accuracy of 97.9%, and it was applied on a Raspberry Pi 4B, with an inference time of 105 ms. The system completed the classification and collection of a single object in 0.88 s. Similarly, in [19], an intelligent waste classification system based on GNet deep learning was generated through Linux and applied on a Raspberry Pi 4B as the main board, in addition to the integrated hardware of a two-degrees-of-freedom servo, touch panel, and camera. The model was trained with the Huawei Garbage Classification Challenge Cup dataset, obtaining an accuracy of 92.62% and an efficiency of 0.63 s, complemented by the development of a graphical user interface based on Python 3.6.8 and QT. Finally, the authors of [20] applied a model in TensorFlow Lite v2.13 on a Raspberry Pi 4 for real-time waste classification and categorization. The servomotors were mounted on a plastic board that separated the waste according to the model's classification and deposited it in appropriate compartments depending on its type.

The present study analyzes the effectiveness of four CNN models in MSW detection, using a Raspberry Pi 4 Model B. YOLOv4-tiny, YOLOv7-tiny, YOLOv8-nano, and YOLOv9-tiny models were trained and evaluated, with their performance compared in terms of precision, inference speed, and resource usage in an embedded system. A dataset was adjusted with 85% of images for training and 15% for testing. In the experiments conducted, learning rate, weight decay, epochs, and online training were configured, generating image rotation and mosaics, and the behaviors of model precision and loss were observed. The comparison of the models allowed for identification of the most suitable one for real-time MSW detection, considering hardware limitations and the need to maintain a balance between efficiency and precision in a low-cost and low-energy-consumption environment. The precision results obtained were as follows: YOLOv4-tiny, 91.71%; YOLOv7-tiny, 91.34%; YOLOv8-nano, 93%; and YOLOv9-tiny, 92%. Their application in an embedded system with an HQ camera obtained an average of 93% CPU usage and an inference time of 1800 ms.

2. Materials and Methods

2.1. Materials

For this research, a Dell G15 5530 laptop with CPU i7 13650HX, GPU RTX 4060 8GB VRAM, and 32 GB RAM, a Raspberry Pi 4 model B with an HQ camera, Roboflow tool [21], YOLOv4 tiny [22], YOLOv7 tiny [23], YOLOv8 nano [24] and YOLOv9 nano [25] were employed.

2.2. Methodology

To evaluate and compare the effectiveness of four convolutional neural network models for MSW detection, a four-phase methodology was developed, as shown in Figure 1.

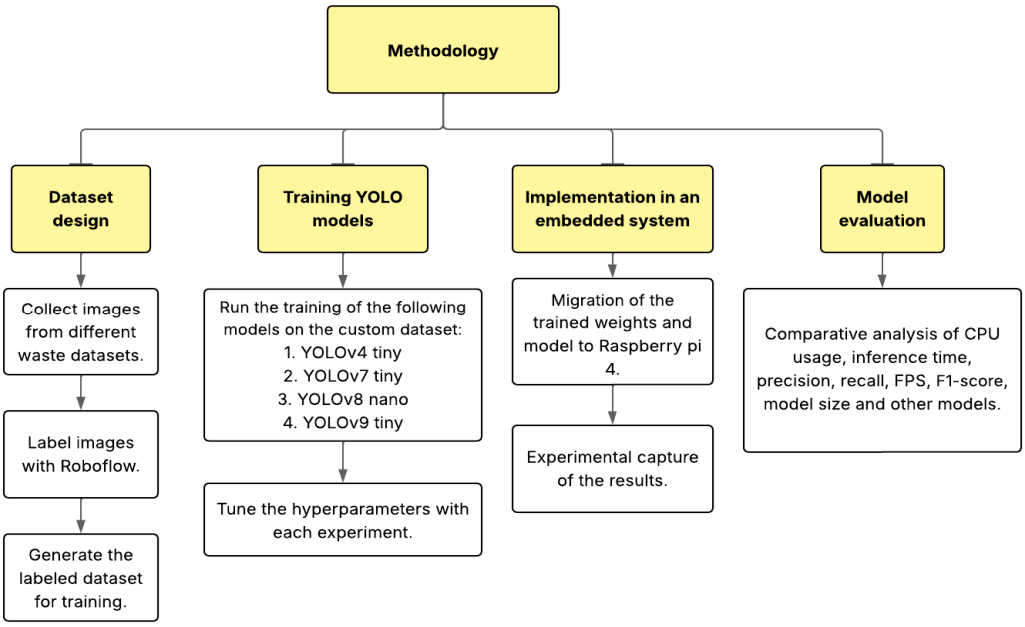


Figure 1. Methodological diagram.

2.2.1. Dataset Design

The design of the dataset of 1883 images involved labeling based on a comparative analysis evaluating the types of waste, as well as image characteristics such as resolution, white backgrounds without irregular edges, uncontrolled backgrounds in green areas, and waste combinations. In addition, several different datasets, including Garbage Classification and Trashnet, were used, which were classified and balanced into inorganic, with 997 images (53%), and organic, with 886 images (47%). For labeling, the Roboflow tool was used, manually delimiting the areas of interest for the objects to be detected. Subsequently, image preprocessing methods were applied to resize the images to 640 × 640 pixels and perform automatic contrast adjustments. Additionally, data augmentation techniques were implemented, rotating the images at angles of +15° and −15°, and applying horizontal and vertical flips to expand the diversity of the dataset, (Table 1, Figure 2).

Table 1. Collected datasets.

Name	No. of Categories	No. of Images	Annotation	Image Type	Author
UAVVaste	1 (garbage waste in green areas)	772	Segmentation	Waste capture in open fields	[26]
Trashnet	6 (glass, paper, cardboard, plastic, metal, general waste)	2527	Classification	Clean background or white background	[27]
Waste Classification data	2 (organic and recyclable objects)	~25,000	Classification	Generated by Google searches	[28]
Garbage Classification	12 (battery, biological, brown glass, cardboard, clothes, green glass, metal, paper, plastic, shoes, trash, white glass)	15,150	Classification/ Detection	Combining the "clothing dataset" and the web scrapping tool	[29]
TrashBox	7 (medical waste, electronic waste, plastics, paper, metal, glass, cardboard)	17,785	Classification/ Detection	Generated by the web	[30]

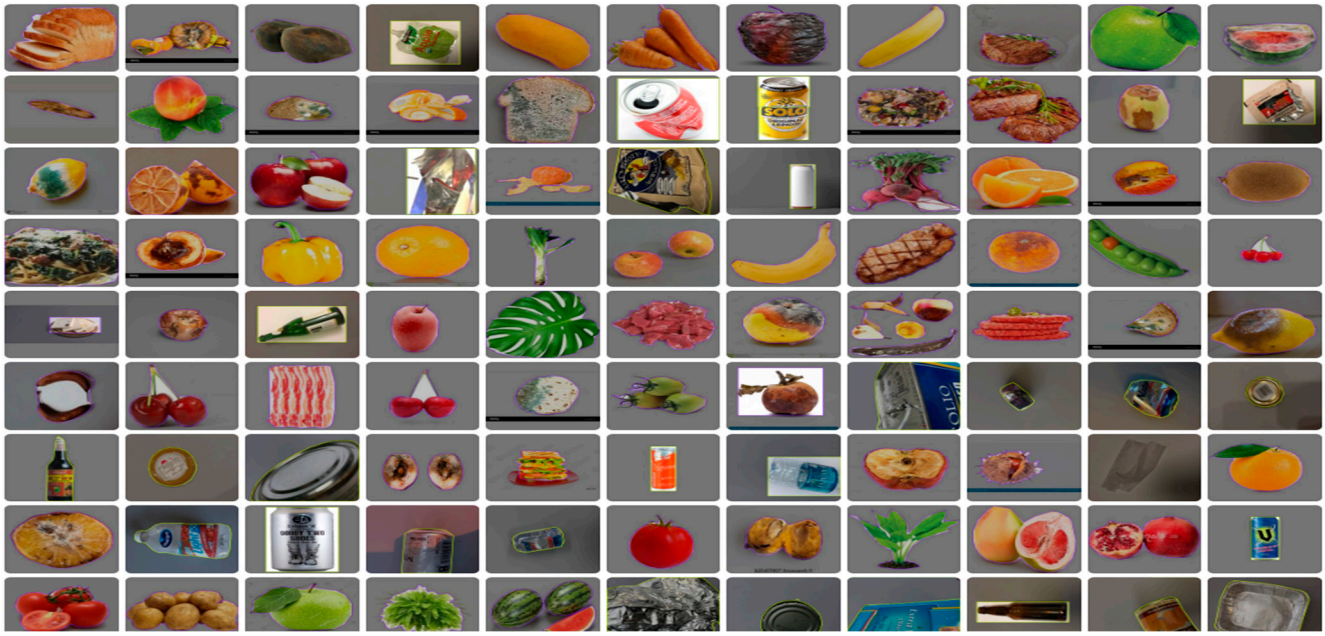


Figure 2. Dataset representation built.

2.2.2. YOLO Model Training

For training the models, the hyperparameters and the custom dataset of 1883 images were considered, using 85% for training and 15% for testing, along with online data augmentation techniques such as flipud: 0.5, fliplr: 0.5, and mosaic: 1.0 (Table 2).

Table 2. Trained models.

Model Settings: YOLOv4 Tiny		**Model Settings: YOLOv7 Tiny**		**Model Settings: YOLOv8 Nano**		**Model Settings: YOLOv9 Tiny**	
Input	640 × 640	Input	640 × 640	Input	640 × 640	Input	640 × 640
Learning rate	0.002	Learning rate	0.001	Learning rate	0.001	Learning rate	0.001
Weight decay	0.0002	Optimizer	Adam	Optimizer	Adam	Optimizer	Adam
Optimizer	Adam	Momentum	0.937	Momentum	0.937	Momentum	0.937
Momentum	0.937	Batchsize	32	Batchsize	32	Batchsize	32
Batchsize	32	Subdivisions	8	Subdivisions	8	Subdivisions	8
Subdivisions	8	Total epoch	100	Total epoch	100	Total epoch	100
Total iterations	6000						

For training YOLO, the input image is divided into a grid of size, where each cell generates a region of interest and predicts the number of bounding boxes, along with its confidence level in each of them, and the number of model classes with their probabilities. The predictions were encoded in a tensor of size S × S (B × 5 + C), as shown in Figure 3 [31], where, S is the number of cells in the image grid; B is the number of bounding boxes; C is the number of model classes; and 5 is the information associated with each bounding box (x, y, w, h, confidence).

The models were adjusted with their own hyperparameters, and their precision, recall, and F1-Score were calculated (Table 3, Figure 4).

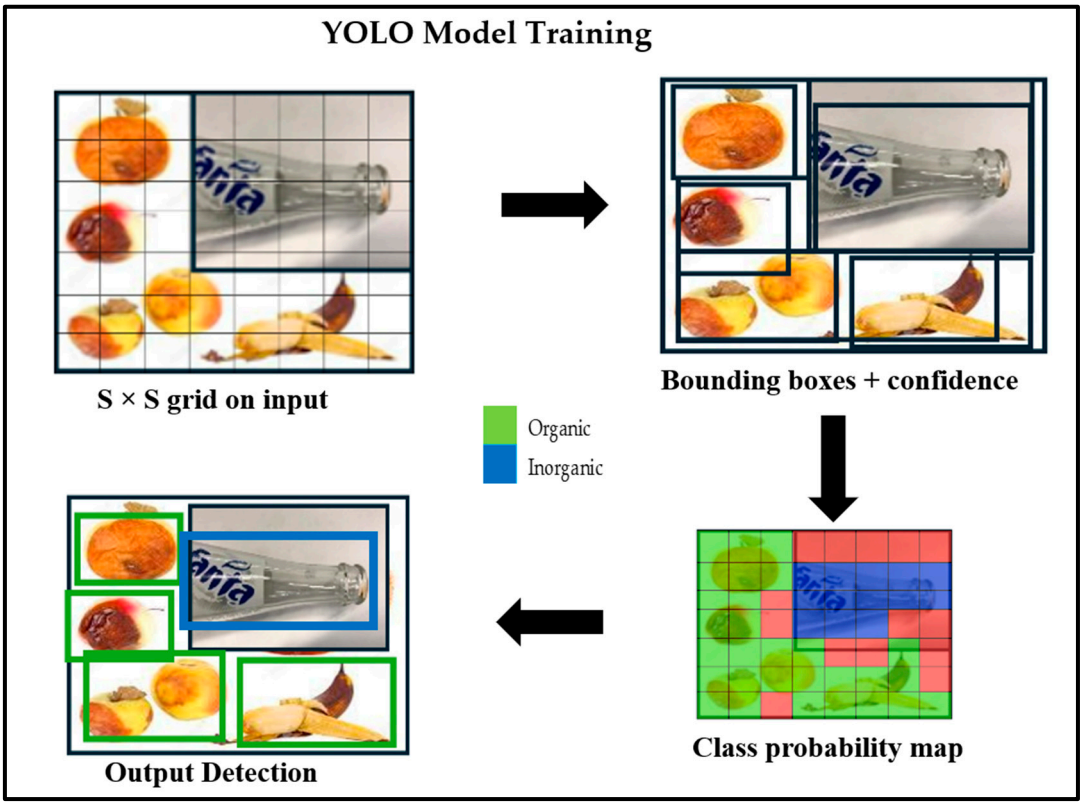


Figure 3. Representation of YOLO model training, adapted from [31].

Table 3. YOLO model calculations.

Model	Clase	Valores	Precision	Recall	F1-Score	IoU
YOLOv9 tiny	Inorganic	TP = 0.97 FN = 0.19 FP = 0.04	$\frac{0.97}{0.97+0.04}$ 0.96	$\frac{0.97}{0.97+0.19}$ 0.836	$2 \times \frac{0.96 \times 0.836}{0.96+0.836}$ $\frac{1.605}{1.796}$ 0.893	$\frac{0.97}{0.97+0.19+0.04}$ $\frac{0.97}{1.2}$ 0.808
	Organic	TP = 0.93 FN = 0.04 FP = 0.06	$\frac{0.93}{0.93+0.06}$ 0.939	$\frac{0.93}{0.93+0.04}$ 0.958	$2 \times \frac{0.939 \times 0.958}{0.939+0.958}$ $\frac{1.799}{1.897}$ 0.948	$\frac{0.93}{0.93+0.04+0.06}$ $\frac{0.93}{1.03}$ 0.903

Validation Metrics

To evaluate the performance of the classification or detection model, the numbers of True Positives (TPs), False Positives (FPs), True Negatives (TNs), and False Negatives (FNs) obtained from the confusion matrices generated in Figure 4 were considered, and calculated in relation to the following:

The Intersection over Union (IoU)

Each input image contains bounding boxes (ground-truth); during the learning process, new predicted bounding boxes are generated. When there is an object within the area of interest, the generated bounding box overlaps with the referenced one, calculating the union area with the obtained confidence, obtaining a relationship between the intersection zone and the union area; see Equation (1) and Tables 3 and 4 [32].

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

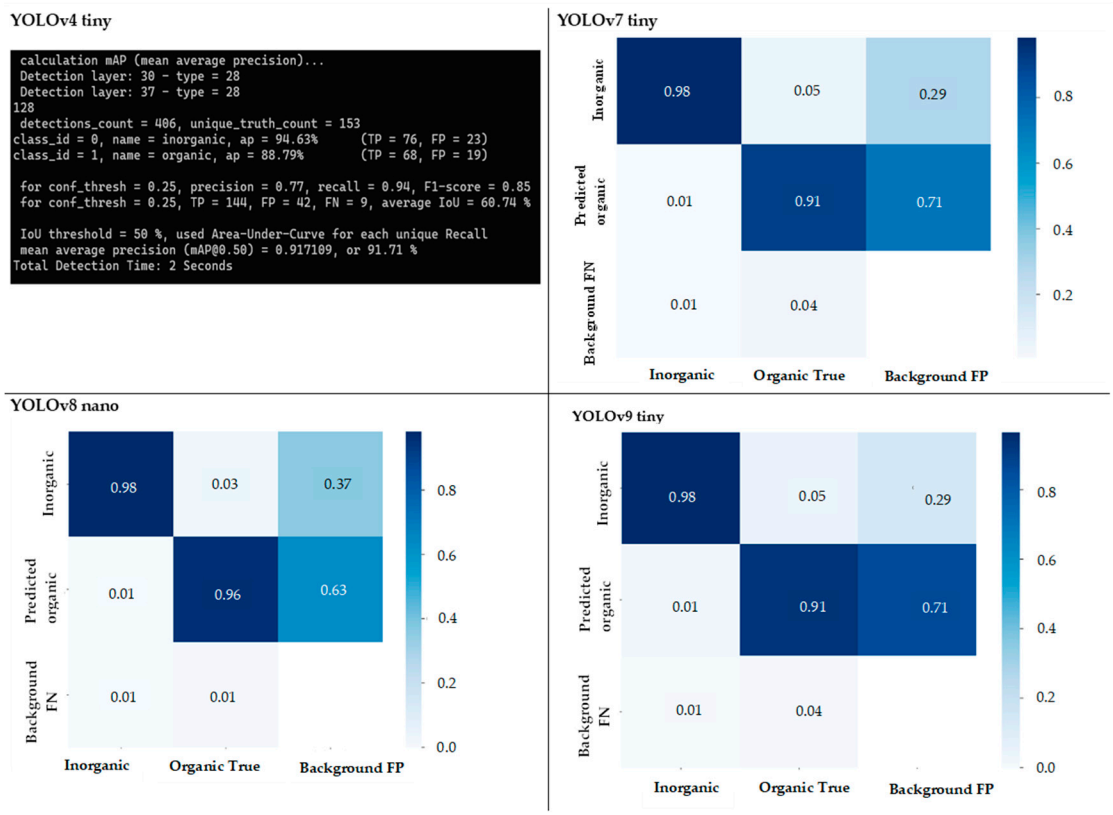


Figure 4. Confusion matrices generated.

Table 4. Metrics obtained from YOLO models.

Model	Precision	Recall	F1-Score	IoU
YOLOv4 tiny	0.91	0.94	0.85	0.60
YOLOv7 tiny	0.91	0.93	0.91	0.83
YOLOv8 nano	0.94	0.97	0.85	0.64
YOLOv9 tiny	0.92	0.94	0.92	0.85

Precision

Precision measures the proportion of correct detections among all detections made. A high precision value indicates that the model is making few false detections; see Equation (2) and Tables 3 and 4 [20].

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

Recall

Recall measures the proportion of real objects that are correctly detected. A high recall value indicates that the model is finding most of the objects present in the images; see Equation (3) and Tables 3 and 4 [33].

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

F1-score

According to Ultralytics, this metric allows for the observation of the balance between *precision* (P) and *recall* (R), evaluating model performance in a more balanced way, identify-

ing whether a model works well across all classes or whether it is biased towards a specific class; see Equation (4) and Tables 3 and 4.

$$F1 - score = 2 \times \frac{P \times R}{P + R} \quad (4)$$

Another important aspect to consider in model training is the graphical representation of metrics to visualize the model's performance during its training (Figure 5).

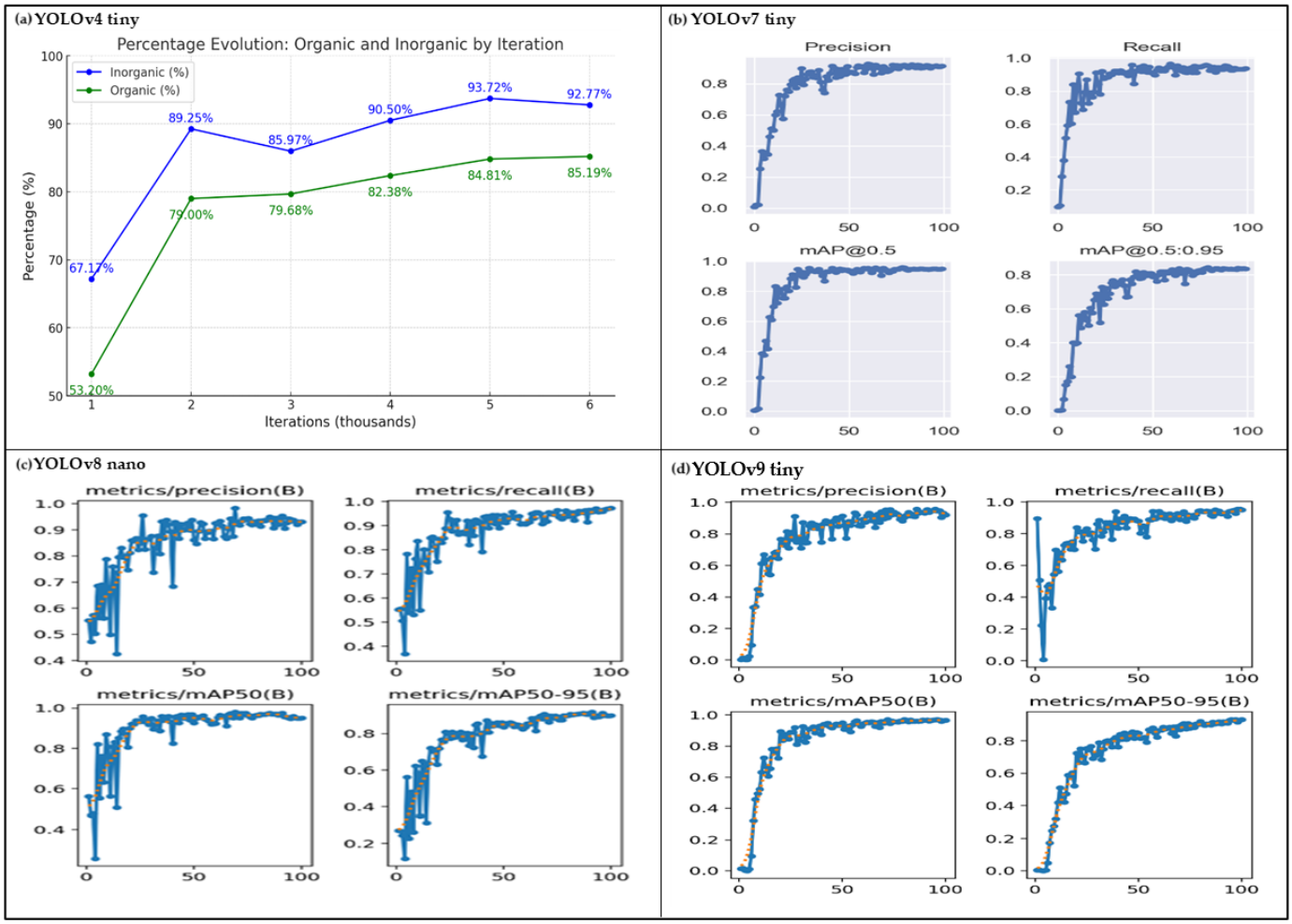


Figure 5. Graphs of models' performance in training.

2.2.3. Embedded System Implementation

During implementation of the embedded system, the trained weights of each model were migrated to the Raspberry Pi 4 Model B device, where each one was executed in a Python 3.13.0 language environment in Thonny IDE to start model detection. Experimental tests were conducted with five organic (general food waste) and inorganic (plastics, paper, glass, aluminum) waste items. This allowed for the identification of their behavior in a real environment.

YOLOv4 tiny: This detected organic and inorganic residues with difficulty, due to lighting and background problems and instability according to the object's position, and it had a Central Processing Unit (CPU) usage of around 80% (Figure 6).

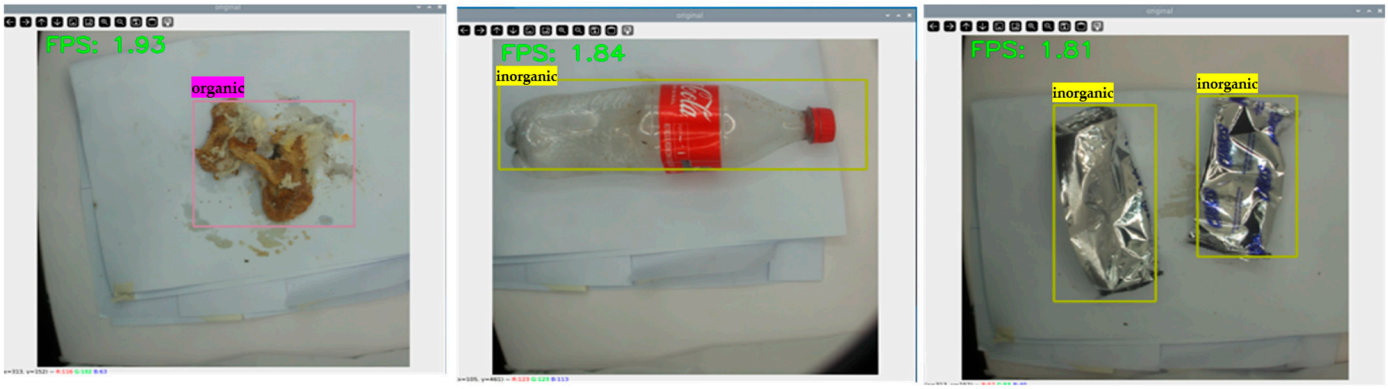


Figure 6. Execution samples of YOLOv4 tiny.

YOLOv7 tiny: This had high precision in recognizing waste, but presented erroneous object detection in empty background areas, as well as having high CPU usage (97%). Its effectiveness was very dependent on the object's location and lighting, and it showed weakness in detecting organic waste at a distance, as well as encountering high wait times in order to perform good inferences (2 s on average) (Figure 7).

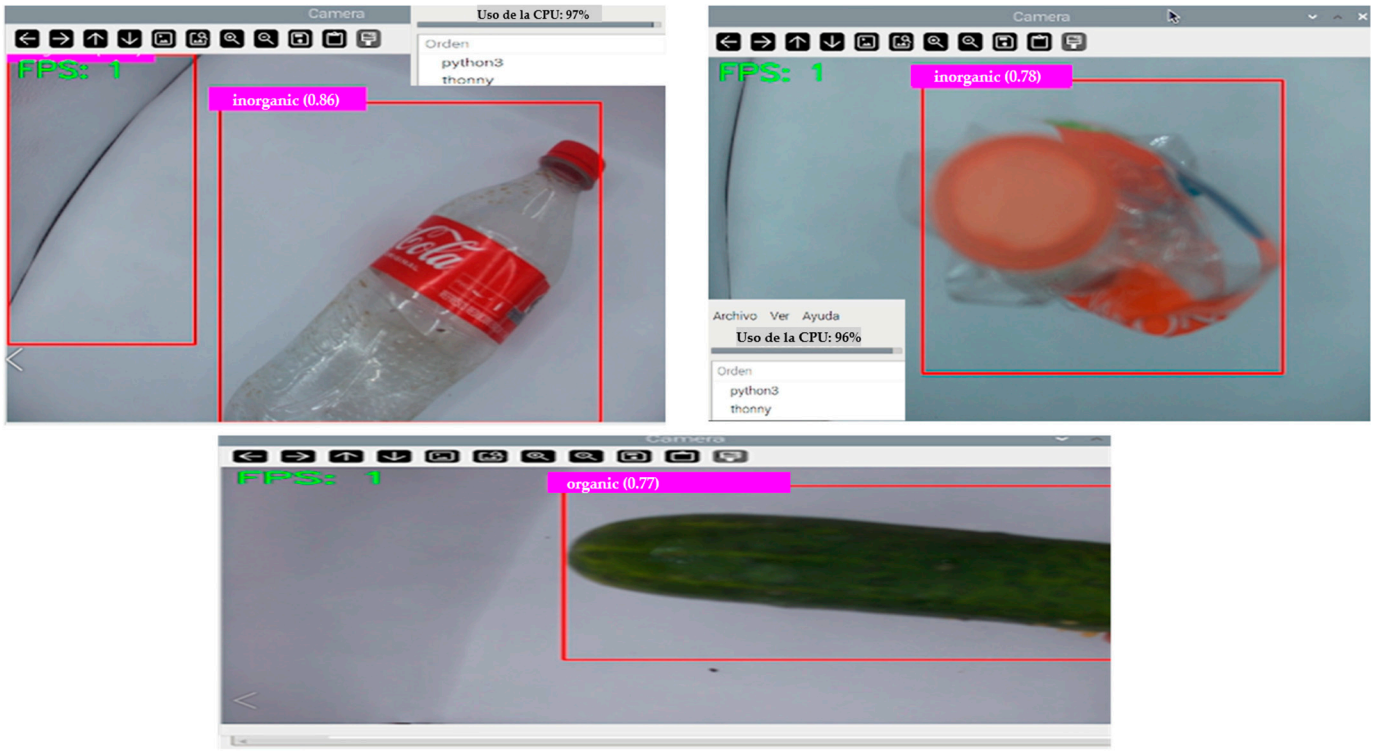


Figure 7. Execution samples of YOLOv7 tiny.

YOLOv8 nano: This made correct class detection, showing dependence on the type of waste in its orientation, and its detection precision was between 40% and 80%. Generally, this model had acceptable performance, with erroneous detection for some objects, generating multiple bounding boxes, and having a CPU usage between 80% and 95% (Figure 8).

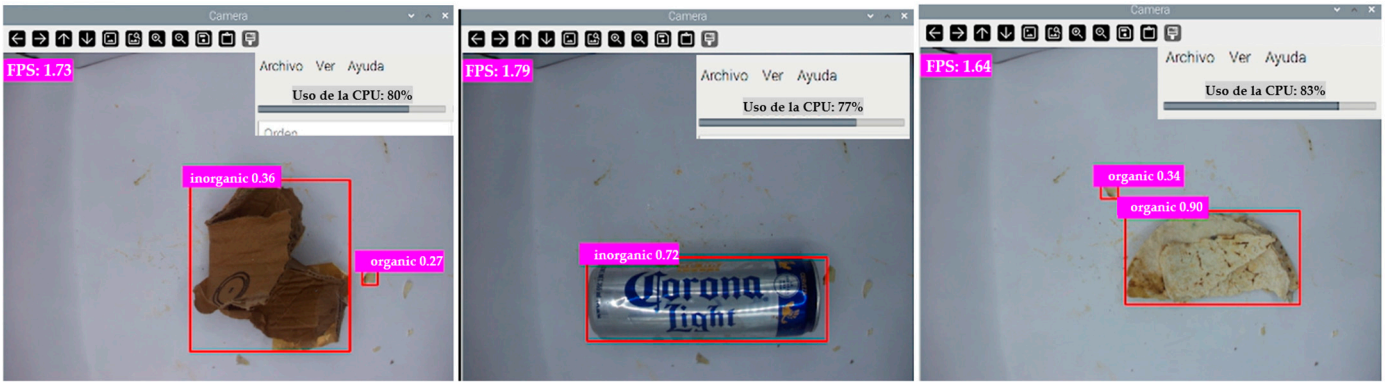


Figure 8. Execution samples of YOLOv8 nano.

YOLOv9 tiny: This had a CPU consumption between 80% and 85%, made correct class detection with a precision above 70%, did not present problems in generating bounding boxes for the test waste set under the same environment and lighting, and had an average inference time of 1800 ms (Figure 9).

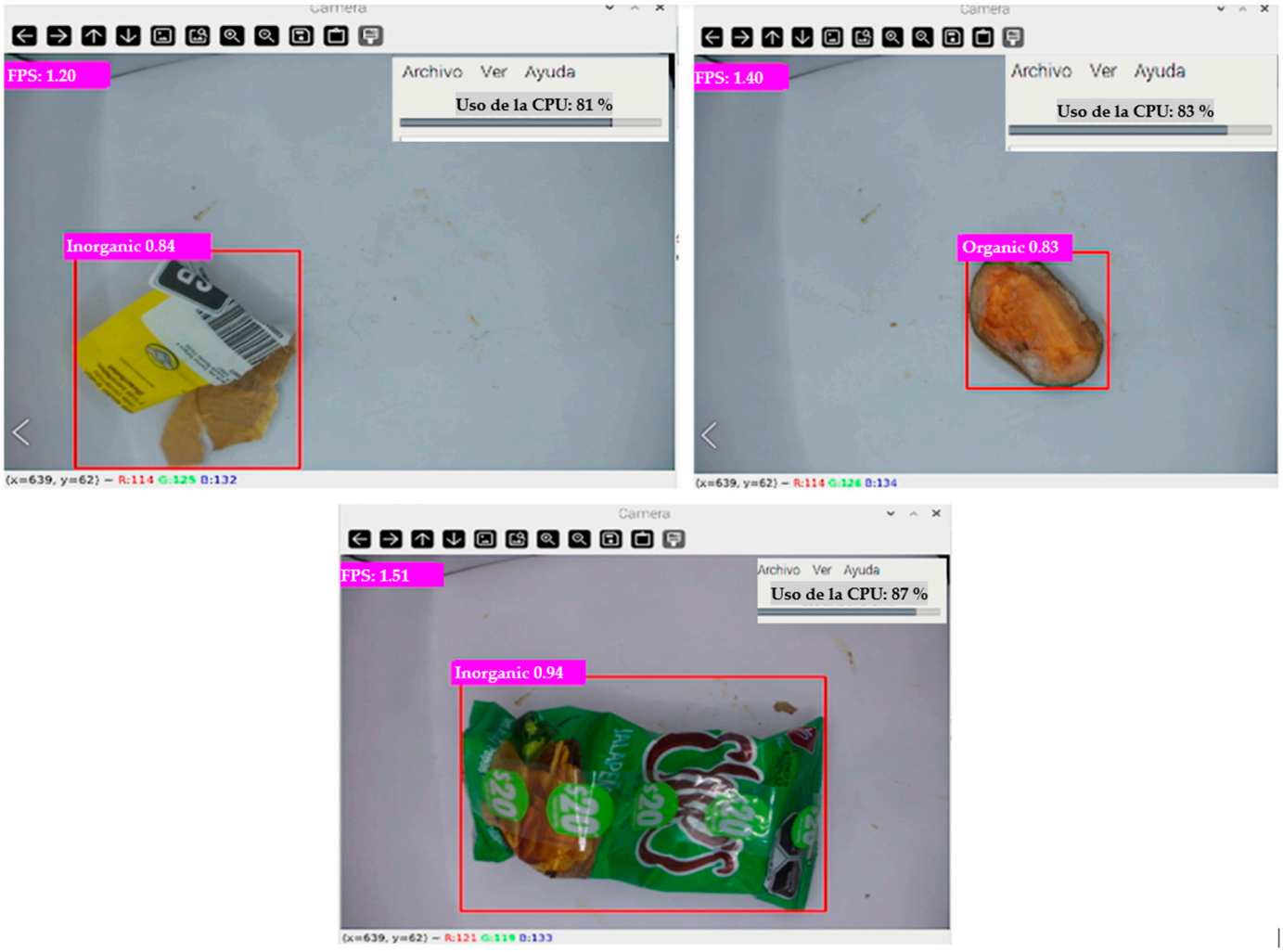


Figure 9. Execution samples of YOLOv9 tiny.

Energy Usage Methods

For the experimentation with the YOLO models implemented on a Raspberry Pi (5 V and 3 A), a 38,600 mAh battery and a 5 V 10 W solar panel were used. The battery capacity in watt-hours was obtained using Equation (5):

$$E = \left(\frac{38600\ \text{mAh}}{1000}\right) \times 3.7\ \text{V} \approx 142.82\ \text{Wh} \quad (5)$$

Regarding the energy consumption of the device, Equation (6) was applied:

$$P = V \times I = 5\ \text{V} \times 3\ \text{A} = 15\ \text{W} \quad (6)$$

where P is power, V is voltage, and I is current.

The theoretical operating time without the use of the solar panel was obtained using Equation (7):

$$T = \left(\frac{E}{P}\right) = \left(\frac{142.82\ \text{Wh}}{15\ \text{W}}\right) \approx 9.52\ \text{h} \quad (7)$$

where T is time, E is battery capacity, and P is power.

The solar panel was tested under favorable conditions for at least four hours, with its solar energy and total energy calculated using Equations (8) and (9), and the total autonomous operating time of the device connected to the battery and solar panel was calculated using Equation (10):

$$E - solar = 4\ \text{h} \times 10\ \text{W} = 40\ \text{Wh} \quad (8)$$

$$E - total = 142.82\ \text{Wh} + 40\ \text{Wh} = 182.82\ \text{Wh} \quad (9)$$

$$T - total = \frac{182.82\ \text{Wh}}{15\ \text{W}} \approx 12\ \text{h} \quad (10)$$

2.2.4. Model Evaluation

Each model was evaluated based on the results observed during its implementation. The experimentation was conducted with the same background, environment, and identical conditions, considering CPU usage, inference time (ms), Frames Per Second (FPS), precision, recall, f1-score, and the size of the generated model. Each waste sample differed from the datasets used during training, and the results observed among other models also differed (Table 5).

Table 5. Comparison of detection models.

Model	Input Resolution	mAP (%)	Model	Input Resolution	mAP (%)
Faster R-CNN [34]	~1000 × 600	73.2	SDD [35]	513 × 513	78.2
SSD512 [34]	512 × 512	76.8	YOLO v4 Tiny	640 × 640	91.7
SSD300 [34]	300 × 300	74.3	YOLO v7 Tiny	640 × 640	94.9
EfficientDet [36]	1536 × 1536	80.4	YOLO v8 Nano	640 × 640	97.6
RetinaNet [37]	600 × 600	81.6	YOLO v9 Tiny	640 × 640	96.8

3. Results

As object detection models for situations with limited resources, YOLO models presented limitations during their implementation in the Raspberry Pi 4 Model B system with a 12.3 MP HQ Camera, with incorrect or false bounding boxes. For example, YOLOv4 tiny was highly dependent on waste illumination, and YOLOv7 tiny had high CPU usage, required close distance, and in some cases, generated empty bounding boxes. YOLOv8 nano correctly detected waste, creating multiple bounding boxes. YOLOv9 tiny had a more stable

CPU consumption, between 80% and 85%, lower detections with an initial accuracy of 70%, and a high inference time (1800 ms) without a notable change when performing detections, compared with YOLOv8 nano (1500 ms). Finally, the YOLOv9 tiny model achieved the best performance, with an average precision of 92.11%, recall of 94.97%, F1-score of 0.729, inference time of 1800 ms, and compact size of 4.43 mb, and in the experimental tests, it did not present errors (Table 6, Figure 10).

Table 6. Performance of models in real time.

Model	CPU Usage	Inference Time (ms)	FPS	Precision	Recall	F1-Score	Model Size (MB)	mAP@0.5
YOLO v4 Tiny	96%	2000	1	91.71%	94%	0.85	22.4	0.917
YOLO v7 Tiny	98%	1900	1	91.34%	93.83%	0.344	11.7	0.949
YOLO v8 Nano	86%	1800	1.70	93%	97.34%	0.665	5.96	0.976
YOLO v9 Tiny	86%	1800	~1.50	92.11%	94.97%	0.729	4.43	0.968

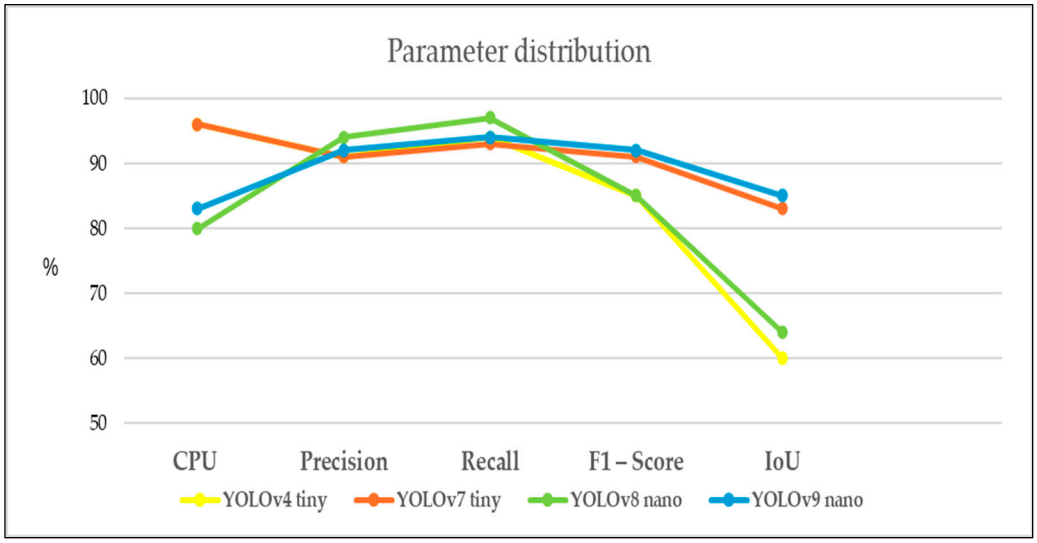


Figure 10. Behavior of YOLO models on Raspberry Pi.

To understand the performance of the models, visual predictions were made with different types of organic and inorganic waste, indicating the confidence level for each class. For YOLOv4 tiny, some low-confidence predictions were observed, such as classifying a plastic bottle as organic and errors in cases of degraded objects like moldy fruit (Figure 11).

Likewise, YOLOv7 tiny had precise detection with high confidence levels above 80%, with adequate detection of materials such as plastic, metal, paper, and organic waste, highlighting the precision in the adjustment of bounding boxes, whereas some waste, like papers and aluminum, presented lower confidence levels (50%) (Figure 12a). YOLOv8 nano had high performance in detecting organic and inorganic waste, with confidence levels above 90%, correctly identifying fresh and decomposing foods, such as fruits, vegetables, and moldy bread, as well as metal plates and other inorganic materials; however, it had false detection in the case of classifying bread as inorganic (Figure 12b). Finally, YOLOv9 tiny obtained high confidence levels (0.9 or higher) in most predictions, indicating that the model had good performance overall (Figure 12c).

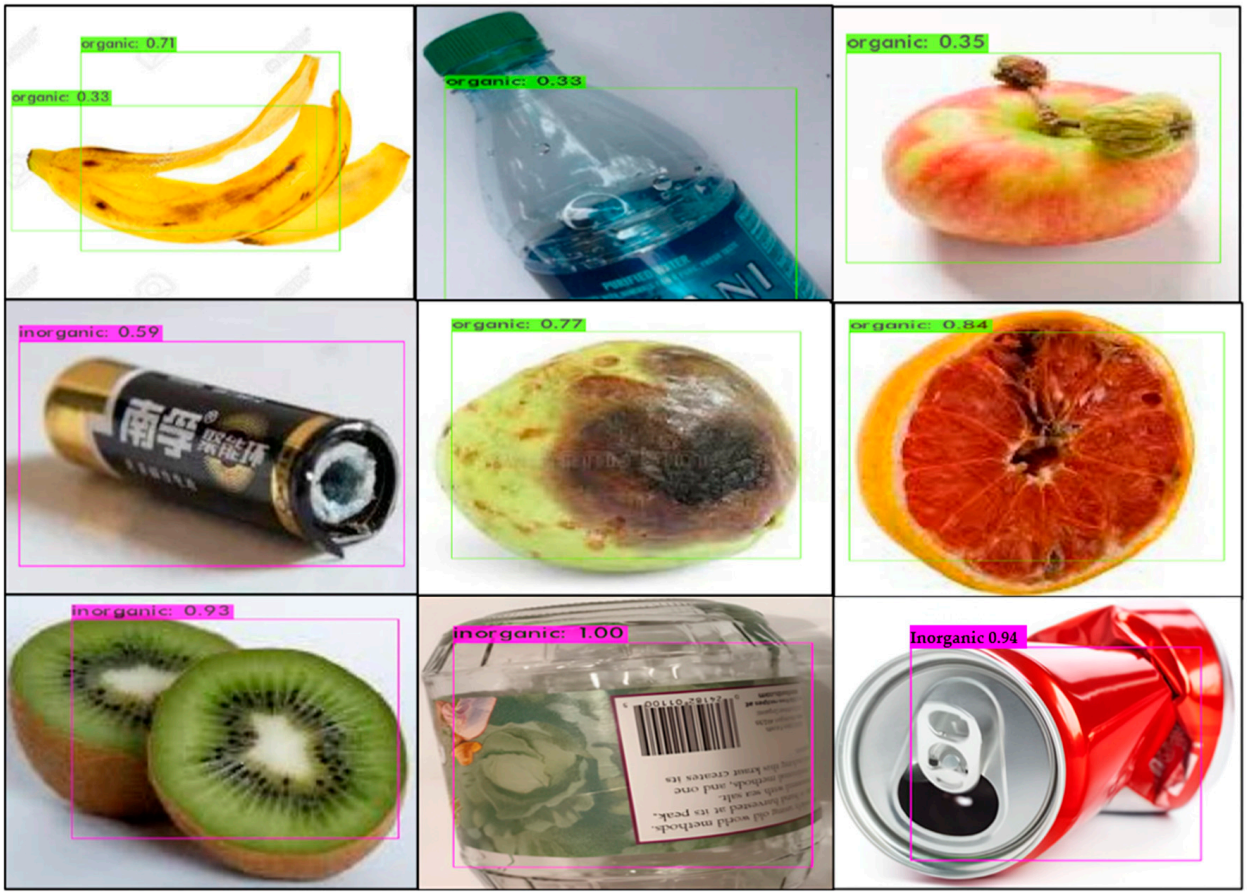
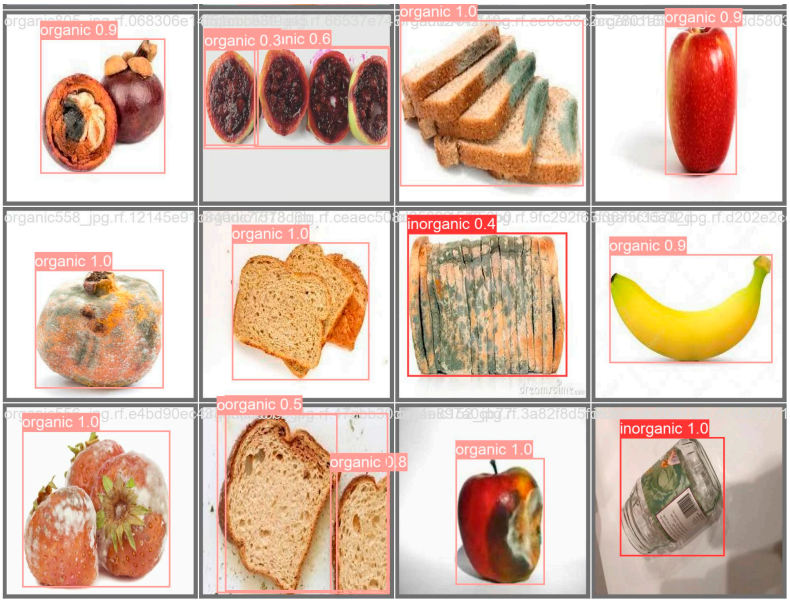


Figure 11. Predictions of YOLOv4 tiny.

(**a**) (**b**)

Figure 12. *Cont*.



(**c**)

Figure 12. Predictions of (**a**) YOLOv7 tiny, (**b**) v8 nano and (**c**) v9 tiny.

3.1. Autonomy of Embedded System

The operation of the embedded system consisted of a battery powered by a 5 V/10 W solar panel that supplied a constant energy source of 5 V/3 A, achieving consistent performance for 12 h, compared to an operating time of 9.52 h without the solar panel.

3.2. Comparison of Detection Models

The comparison of YOLO with other models for object detection in terms of mean Average Precision (mAP) showed a significant improvement, as seen in Figure 13.

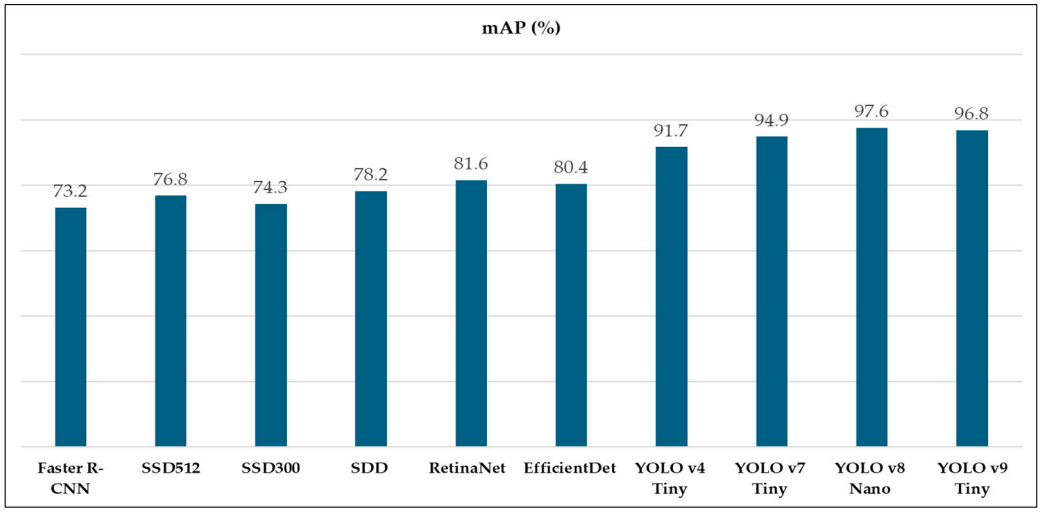


Figure 13. Distribution of mAP (%) across different models.

4. Discussion

Choosing an optimal model for the detection of MSW with compact characteristics, based on an embedded system using a Raspberry Pi 4 Model B minicomputer, requires evaluating the results obtained during real-time testing. This was the case in [38], where the models EfficientDet-D1, SSD ResNet-50 V1, Faster R-CNN ResNet-101 V1, CenterNet ResNet-101 V1, and YOLOv5M were compared through 25 tests using waste images not

included in the training set. This differs from our study, where we used real-time images with four YOLO models in their most compact versions. Another important aspect is the evaluation using mAP@0.5, where [38] obtained a maximum average of 0.613 in their YOLOv5m model, which is lower than the results obtained in this research: YOLOv7-tiny = 0.949, YOLOv8-nano = 0.976, and YOLOv9-tiny = 0.968. Additionally, ref. [38] did not conduct tests on an embedded system. However, refs. [14–16] developed a lightweight CNN model with an efficient algorithm on an embedded device for detecting surface defects in industrial products, achieving accuracy comparable to state-of-the-art models; this aligns with our work on the implementation of lightweight YOLO models in embedded systems. Regarding the dataset, it agrees with [2], in which Trash-net and Public GarbageNet were also used to train the MSW detection model. It is worth mentioning that [2] evaluated a single model with four datasets, while in this research, four models were evaluated with a combined dataset of Trash-net and Public GarbageNet. The use of a Raspberry Pi 4B device as the main board for object detection in embedded systems [6] is consistent with this work. Regarding inference times, in [6], GCNet, MobileNet v2, and ResNet50 were evaluated, obtaining times of 200 ms, 200 ms, and 600 ms, respectively, which are different from the 1800 ms obtained in this experimentation; however, the models in this study detected more than one type of waste simultaneously, while those in [6] only detected a single type of waste.

5. Conclusions

YOLO versions of object detection models require high-spec devices, without considering resource consumption in their implementation; they are commonly not applied in embedded systems, such as the Raspberry Pi 4 Model B with the 12.3 MP HQ camera, due to their limited CPU and GPU capabilities. Regarding the comparison of the models in their different versions, the performance of object detection models has been investigated. This research presents a comparative analysis of four YOLO models tested under the same environmental conditions and with the same datasets, the YOLOv4-tiny, YOLOv7-tiny, YOLOv8-nano, and YOLOv9-tiny versions, to determine each one's performance in terms of CPU usage, inference time, FPS, precision, recall, mAP@0.5, F1-score, and model size. The results obtained show that YOLOv9-tiny achieved a very solid performance in the experimental tests, with an average precision of 92.11%, a recall of 94.97%, an F1-score of 0.729, a compact size of 4.43 MB, and an absence of errors during testing, positioning it as the best model in terms of stability and overall performance. Meanwhile, YOLOv7 demonstrated high precision in recognizing waste, but also detected non-existent objects, exhibited high CPU usage (97%), showed dependence on the object's location and lighting, showed weakness in determining distances with organic waste, and had long inference times for object type identification. This research will allow us to evaluate the behavior of lightweight YOLO models applied on a Raspberry Pi 4B device, without relying on remote systems, for implementation, for example, in an intelligent container for classifying organic and inorganic waste, which would ensure proper management and contribute to the Sustainable Development Goals, specifically Goal 13: Climate Action.

Author Contributions: Conceptualization, C.M.-M., W.C.-F., C.M.-D., Y.E.-A. and D.E.G.-V.; Methodology, M.C.-B. and C.M.-D.; Software, D.B.R.-P., W.C.-F. and Y.E.-A.; Validation, M.C.-B., D.B.R.-P. and C.M.-D.; Formal analysis, M.C.-B., D.B.R.-P. and D.E.G.-V.; Investigation, M.C.-B., D.B.R.-P., C.M.-M. and C.V.M.-V.; Data curation, D.B.R.-P.; Writing—original draft, M.C.-B. and D.B.R.-P.; Writing—review & editing, M.C.-B., D.B.R.-P., C.V.M.-V. and C.M.-D.; Visualization, D.E.G.-V.; Supervision, M.C.-B. and C.M.-M.; Project administration, C.M.-M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data associated with the study will be available in the repository of the National Technological Institute of Mexico/Technological Institute of Chilpancingo, https://rinacional.tecnm.mx/handle/TecNM/149 (accessed on 1 February 2025). The data presented in this study are available upon request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Jin, B.; Li, W. Spatial Effects and Driving Factors of Consumption Upgrades on Municipal Solid Waste Eco-Efficiency, Considering Emission Outputs. *Sustainability* **2025**, *17*, 2356. [CrossRef]
2. Zhou, Y.; Wang, Z.; Zheng, S.; Zhou, L.; Dai, L.; Luo, H.; Zhang, Z.; Sui, M. Optimization of Automated Gar-bage Recognition Model Based on ResNet-50 and Weakly Supervised CNN for Sustainable Urban Development. *Alex. Eng. J.* **2024**, *108*, 415–427. [CrossRef]
3. De Carolis, B.; Ladogana, F.; Macchiarulo, N. YOLO TrashNet: Garbage Detection in Video Streams. In Proceedings of the 2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS), Bari, Italy, 27–29 May 2020; pp. 1–7.
4. Falaschetti, L.; Manoni, L.; Palma, L.; Pierleoni, P.; Turchetti, C. Embedded Real-Time Vehicle and Pedestrian Detection Using a Compressed Tiny YOLO v3 Architecture. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 19399–19414. [CrossRef]
5. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
6. Wang, L.; Ji, W.; Wang, G.; Feng, Y.; Du, M. Intelligent Design and Optimization of Exercise Equipment Based on Fusion Algorithm of YOLOv5-ResNet 50. *Alex. Eng. J.* **2024**, *104*, 710–722. [CrossRef]
7. Chen, Z.; Yang, J.; Chen, L.; Jiao, H. Garbage Classification System Based on Improved ShuffleNet V2. *Resour. Conserv. Recycl.* **2022**, *178*, 106090. [CrossRef]
8. Sachin, C.; Manasa, N.; Sharma, V.; AA, N.K. Vegetable Classification Using You Only Look Once Algorithm. In Proceedings of the 2019 International Conference on Cutting-edge Technologies in Engineering (ICon-CuTE), Uttar Pradesh, India, 14–16 November 2019; pp. 101–107.
9. Liu, S.; Jin, Y.; Ruan, Z.; Ma, Z.; Gao, R.; Su, Z. Real-Time Detection of Seedling Maize Weeds in Sustainable Agriculture. *Sustainability* **2022**, *14*, 15088. [CrossRef]
10. Ultralytics Ultralytics Home. Available online: https://docs.ultralytics.com/es#how-can-i-train-a-custom-yolo-model-on-my-dataset (accessed on 22 March 2025).
11. Wu, Z.; Zhang, D.; Shao, Y.; Zhang, X.; Zhang, X.; Feng, Y.; Cui, P. Using YOLOv5 for Garbage Classification. In Proceedings of the 2021 4th International Conference on Pattern Recognition and Artificial Intelligence (PRAI), Yibin, China, 20–22 August 2021; pp. 35–38.
12. Zeng, M.; Lu, X.; Xu, W.; Zhou, T.; Liu, Y. PublicGarbageNet: A Deep Learning Framework for Public Garbage Classification. In Proceedings of the 2020 39th Chinese Control Conference (CCC), Shenyang, China, 27–29 July 2020; pp. 7200–7205.
13. Yang, Z.; Xia, Z.; Yang, G.; Lv, Y. A Garbage Classification Method Based on a Small Convolution Neural Network. *Sustainability* **2022**, *14*, 14735. [CrossRef]
14. Zhang, D.; Hao, X.; Wang, D.; Qin, C.; Zhao, B.; Liang, L.; Liu, W. An Efficient Lightweight Convolutional Neural Network for Industrial Surface Defect Detection. *Artif. Intell. Rev.* **2023**, *56*, 10651–10677. [CrossRef]
15. Zhang, D.; Wang, Y.; Meng, L.; Yan, J.; Qin, C. Adaptive Critic Design for Safety-Optimal FTC of Unknown Nonlinear Systems with Asymmetric Constrained-Input. *ISA Trans.* **2024**, *155*, 309–318. [CrossRef]
16. Zhang, D.; Hao, X.; Liang, L.; Liu, W.; Qin, C. A Novel Deep Convolutional Neural Network Algorithm for Surface Defect Detection. *J. Comput. Des. Eng.* **2022**, *9*, 1616–1632. [CrossRef]
17. Majchrowska, S.; Mikołajczyk, A.; Ferlin, M.; Klawikowska, Z.; Plantykow, M.A.; Kwasigroch, A.; Majek, K. Deep Learning-Based Waste Detection in Natural and Urban Environments. *Waste Manag.* **2022**, *138*, 274–284. [CrossRef] [PubMed]
18. Laksono, P.W.; Anisa, A.; Priyandari, Y. Deep Learning Implementation Using Convolutional Neural Network in Inorganic Packaging Waste Sorting. *Frankl. Open* **2024**, *8*, 100146. [CrossRef]
19. Fu, B.; Li, S.; Wei, J.; Li, Q.; Wang, Q.; Tu, J. A Novel Intelligent Garbage Classification System Based on Deep Learning and an Embedded Linux System. *IEEE Access* **2021**, *9*, 131134–131146. [CrossRef]

20. Gude, D.K.; Bandari, H.; Challa, A.K.R.; Tasneem, S.; Tasneem, Z.; Bhattacharjee, S.B.; Lalit, M.; Flores, M.A.L.; Goyal, N. Transforming Urban Sanitation: Enhancing Sustainability through Machine Learning-Driven Waste Processing. *Sustainability* **2024**, *16*, 7626. [CrossRef]
21. Roboflow. Available online: https://app.roboflow.com/login (accessed on 24 October 2024).
22. YOLOv4. Available online: https://github.com/AlexeyAB/darknet (accessed on 22 November 2024).
23. YOLOv7. Available online: https://github.com/WongKinYiu/yolov7 (accessed on 22 November 2024).
24. YOLOv8. Available online: https://github.com/ultralytics/ultralytics (accessed on 24 November 2024).
25. YOLOv9. Available online: https://github.com/WongKinYiu/yolov9 (accessed on 24 November 2024).
26. Kraft, M.; Piechocki, M.; Ptak, B.; Walas, K. Autonomous, Onboard Vision-Based Trash and Litter Detection in Low Altitude Aerial Images Collected by an Unmanned Aerial Vehicle. *Remote Sens.* **2021**, *13*, 965. [CrossRef]
27. Thung, G.; Yang, M. *Classification of Trash for Recyclability Status*; CS229; Stanford University: Stanford, CA, USA, 2016.
28. Sashaank Sekar Waste Classification Data. Available online: https://www.kaggle.com/datasets/techsash/waste-classification-data/data (accessed on 9 December 2024).
29. Mostafa Mohamed Garbage Classification (12 Classes). Available online: https://www.kaggle.com/datasets/mostafaabla/garbage-classification (accessed on 9 December 2024).
30. Kumsetty, N.V.; Bhat Nekkare, A.; Kamath, S.S.; Kumar, M.A. TrashBox: Trash Detection and Classification Using Quantum Transfer Learning. In Proceedings of the 2022 31st Conference of Open Innovations Association (FRUCT), Helsinki, Finland, 27–29 April 2022; pp. 125–130.
31. Redmon, J.; Divvala, S.K.; Girshick, R.B.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 11–15 June 2015. [CrossRef]
32. Kim, K.; Kim, K.; Jeong, S. Application of YOLO v5 and v8 for Recognition of Safety Risk Factors at Construction Sites. *Sustainability* **2023**, *15*, 15179. [CrossRef]
33. Gao, X.; Wang, G.; Qi, J.; Wang, Q.; Xiang, M.; Song, K.; Zhou, Z. Improved YOLO v7 for Sustainable Agriculture Significantly Improves Precision Rate for Chinese Cabbage (Brassica Pekinensis Rupr.) Seedling Belt (CCSB) Detection. *Sustainability* **2024**, *16*, 4759. [CrossRef]
34. Sallang, N.C.A.; Islam, M.T.; Islam, M.S.; Arshad, H. A CNN-Based Smart Waste Management System Using TensorFlow Lite and LoRa-GPS Shield in Internet of Things Environment. *IEEE Access* **2021**, *9*, 153560–153574. [CrossRef]
35. Fu, C.-Y.; Liu, W.; Ranga, A.; Tyagi, A.; Berg, A.C. DSSD: Deconvolutional Single Shot Detector. *arXiv* **2017**, arXiv:1701.06659. [CrossRef]
36. Tan, M.; Pang, R.; Le, Q. V EfficientDet: Scalable and Efficient Object Detection. *arXiv* **2019**, arXiv:1911.09070. [CrossRef]
37. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. *arXiv* **2017**, arXiv:1708.02002. [CrossRef]
38. Patel, D.; Patel, F.; Patel, S.; Patel, N.; Shah, D.; Patel, V. Garbage Detection Using Advanced Object Detection Techniques. In Proceedings of the 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), Coimbatore, India, 25–27 March 2021; pp. 526–531.