Smart Recycling: an Object Detection Model for Automated and Efficient Waste Sorting

Francesco Barcherini
Dep. of Information Engineering
University of Pisa
f.barcherini@studenti.unipi.it

Andrea Di Matteo
Dep. of Information Engineering
University of Pisa
a.dimatteo5@studenti.unipi.it

***Abstract*—The rapid growth of municipal solid waste—projected to reach 3.4 billion t/yr by 2050—demands increasingly automated and efficient collection and sorting solutions. Recent systematic reviews of artificial-intelligence approaches show that state-of-the-art object-detection models, particularly the YOLO families, have boosted classification speed and accuracy, yet still suffer from imbalanced datasets, cluttered backgrounds, and limited attention to hazardous materials [citazione qui]. Addressing these limitations, this article introduces Smart Recycling, an end-to-end framework that pairs YOLOv11-Detect with RTDETRv2, fine-tuned via k-fold cross-validation on Yolo Waste Detection Dataset hosted on Roboflow Platform covering five macro-categories (glass, metal, organic, paper, plastic). The system adds a novel capability: spatial clustering of homogeneous objects to optimise robotic grasping exploiting DBSCAN algorithm. Metrics taken into account have been mAP@50, precision, recall, F1-score, confusion matrix and clustering-based mAP@50. Preliminary experiments yield a mean 0.8 mAP@50 while sustaining approximately 20 FPS on NVIDIA Tesla T4 desktop GPU.**

***Index Terms*—waste detection, data augmentation, deep learning, YOLO, RT-DETR, real-time detection, robotic grasping, DBSCAN, clustering**

I. Introduction

The World Bank projects that global municipal solid waste will increase from 2.1 billion tonnes per year in 2023 to approximately 3.4 billion tonnes by 2050, placing growing pressure to adopt automated waste sorting solutions [8]. Deep learning–based detectors, particularly those in the YOLO family, have become the main approach in waste classification research due to their real-time inference speed and easy to deploy capabilities [10], [11]. However, recent surveys highlight two persistent challenges: severe class imbalance and limited robustness in complex, cluttered scenes [8], [9]. These limitations motivate the development of a new processing pipeline that integrates detection models with a grasp-aware post-processing module capable of identifying object clusters suitable for single-grasp operations.

A. Fold-aware augmentation

Two complementary augmentation techniques were applied on the training set: (i) classical photometric and geometric transformations (e.g., flips, rotations, exposure shifts), and (ii) synthetic generation combining copy–paste and rendered scenes. Synthetic objects, extracted from real segmentation masks, are either pasted onto real backgrounds or composited into generated backgrounds, while preserving key statistical properties of the training set (e.g., per-image objects average, bounding-box center position distribution, bounding-box geometry). This strategy not only achieves a fully balanced class distribution but has also shown to improve mAP@50 by up to three percentage points in class-imbalanced settings [21].

B. Available models

We benchmark five YOLOv11-Detect variants (n, s, m, l, x) and two RT-DETR v2 scales (l, x) using identical k-fold splits. Literature reviews predict the following trend: larger backbones increase mAP but cut inference speed—e.g., YOLOv5n→YOLOv5x trades +12 mAP for a $\times 4$ latency hike, while RT-DETR-X lifts AP by $\sim$2 pp over RT-DETR-L yet drops from 114 FPS to 74 FPS [10], [12], [13]. On the other hand, increasing the size of the models can lead to overfitting and to lack of generalization capabilities.

C. Cluster-level evaluation with DBSCAN

Beyond per-box metrics, we post-process predictions with DBSCAN to group spatially close items and offer graspable clusters to the robotic picker. Under fixed clustering parameters, *cluster mAP@50* rises linearly with detector mAP, confirming that higher box-level quality directly translates into multi-object pick success.

D. Summary

This paper introduces *Smart Recycling*, a pipeline that (i) compares YOLOv11-Detect with RT-DETR v2 for waste detectio, (ii) counteracts class imbalance via augmentation and synthetic generation, and (iii) evaluates grasp feasibility through a novel cluster-level metric. Experiments delved on NVIDIA T4 show that YOLOv11-s + RT-DETR-L deliver a mean $> 80\%$ mAP@50 while sustaining up to 30 FPS, satisfying real-time constraints for automated waste sorting.

II. Related Work

A. Literature Reviews

Recent reviews confirm that deep learning–based object detection has become the standard paradigm for automated waste sorting. Fotovvatikhah *et al.* [8] analysed 97 publications from 2020 to 2025 and found that over 70 % employed Convolutional Neural Networks (CNNs) primarily YOLO, while

dataset imbalance and background complexity were cited as major unresolved issues. Earlier, Umer *et al.* [9] reviewed smart-bin applications and highlighted the scarcity of real-time benchmarks, emphasising the need for models that generalise effectively dynamic environments.

B. Public Datasets and Annotation Quality

Over fifteen public datasets are available for waste detection, including TrashNet, TACO, WaDaBa, OpenLitterMap, and Domestic-Trash. While these datasets support benchmarking, they consistently suffer from two critical limitations: (i) severe class imbalance, where common materials such as plastic and paper dominate, while hazardous or fragile items like glass shards remain underrepresented; and (ii) inconsistent taxonomies, with label counts ranging from 2 to over 150, which makes cross-dataset transferability and model generalization difficult.

C. You Look Only Once (YOLO)

a) Architecture improvements: Several YOLO variants have been tailored for waste detection. **Skip-YOLO** introduces wide kernels and dense skip connections, improving performance by over 20 percentage points (mAP) compared to YOLOv3 in multi-scene datasets. **YOLOv8 + CBAM** integrates spatial attention to handle class imbalance more effectively, achieving nearly 90 % mAP across 17 waste categories while maintaining real-time inference on edge devices.

b) Benchmark comparisons: In comparative studies on e-waste, YOLOv8 outperforms YOLOv5 and YOLOv7 in both precision and latency. Although, Mask R-CNN remains slightly more accurate for instance segmentation, but YOLOv8 achieves roughly three times the frame rate.

c) Edge deployment and smart-city use cases: YOLO pipelines are increasingly adopted in smart waste bins. A two-stage YOLOv5→YOLOv8 cascade reliably detects overflowing containers and misplaced items, though certain categories (e.g., wet waste) still underperform due to visual ambiguity and limited training data.

D. Transformer-Based Detectors (RT-DETR)

Transformer-based detectors such as RT-DETR combine global context modeling with near-YOLO inference speed, making them promising candidates for real-time waste detection tasks.

a) Street-level and roadside waste: **RGD-DETR** enhances the baseline with state-space interaction modules and a sorting-aware decoder, achieving a +2 pp gain in mAP on occluded roadside litter datasets.

b) Aquatic environments: **RTDETR-MARD** leverages multi-scale adaptive fusion to better detect small debris, achieving 90 % mAP@50 at 32 FPS on the NVIDIA AGX Orin. **LR-DETR** reduces computational cost by nearly 25 % while improving mAP by five points in river-litter detection benchmarks.

c) Electronic waste and PCB dismantling: Specialised RT-DETR adaptations for printed circuit board (PCB) recycling outperform recent YOLO variants in component-level detection, while maintaining inference speeds above 25 FPS on standard desktop GPUs.

E. Two-Stage Detectors

Despite their slower inference speed, two-stage architectures remain competitive in tasks where instance segmentation is critical. For example, a Faster R-CNN implementation running on Android achieves 98 % mean accuracy in waste classification, though it requires approximately one second per image on mid-range mobile hardware.

III. Backgrounds

A. System Architecture

The proposed system architecture in Figure 1 is designed as a full end-to-end framework for automated waste detection, classification, and spatial clustering to support collection tasks. The architecture follows a sequential pipeline composed of five principal phases: (i) data pre-processing and augmentation; (ii) model training using deep object detectors; (iii) model selection through cross-validation; (iv) inference on new input images; and (v) cluster detection.

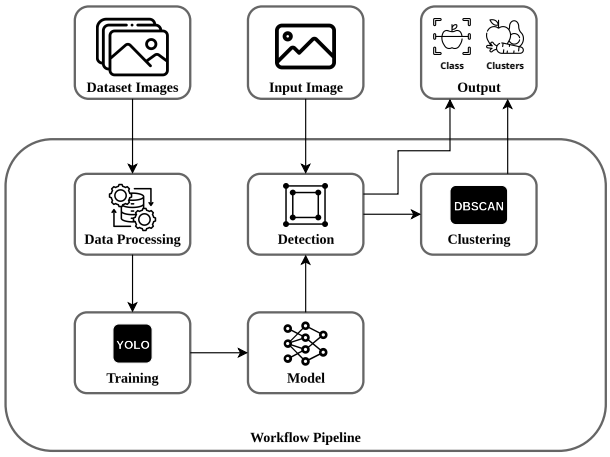

Fig. 1. Functional organization of the intelligent system

In the first phase, raw images from the dataset undergo a structured pre-processing workflow aimed at enhancing the model's generalization capacity and correcting for class imbalance. This includes standard geometric transformations such as horizontal and vertical flips, rotations at fixed angles and controlled exposure shifts. Additionally, a synthetic image generation pipeline is implemented based on the TACO dataset, which provides pixel-accurate segmentation masks. These segmented instances are remapped to five macro-classes (glass, metal, organic, paper, plastic) and composited either into real backgrounds or into procedurally generated synthetic environments.

The second phase involves the training of pre-trained object detection models, namely YOLOv11-Detect and RT-DETR v2,

both tailored for real-time inference and high precision in cluttered visual environments. The training pipeline integrates a stratified k-fold cross-validation procedure ($k = 5$).

Once derived, the models are employed to compute a refined value of mean Average Precision at Intersection over Union 0.5 threshold (mAP@50) on unseen input images. The system performs both detection and classification of individual waste objects, assigning each detected bounding box a category and a confidence score.

The system aims not only to identify discrete objects, but also to detect and group spatial clusters of homogeneous waste items. This is achieved by applying the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm to the spatial coordinates of the predicted bounding boxes. By clustering objects belonging to the same class and located in close proximity, the system enables, for example, a robotic manipulator to collect grouped items in a single grasping operation, thereby significantly improving efficiency.

Finally, the system produces two distinct outputs. The first is a standard list of bounding boxes, each annotated with the predicted class and associated confidence score, suitable for traditional object-level sorting. The second output consists of aggregated bounding boxes encompassing clusters of same-category objects. Both outputs are evaluated during the test phase: the per-object detections are assessed via standard mAP, precision and recall metrics, while the clustering results are compared against ground-truth cluster formations to compute a cluster-level mAP@50, measuring the quality of spatial aggregation.

B. YOLOv11 Architecture

The YOLOv11 (You Only Look Once) architecture, illustrated in Figure 2, is logically structured into three main components: Backbone, Neck, and Head.

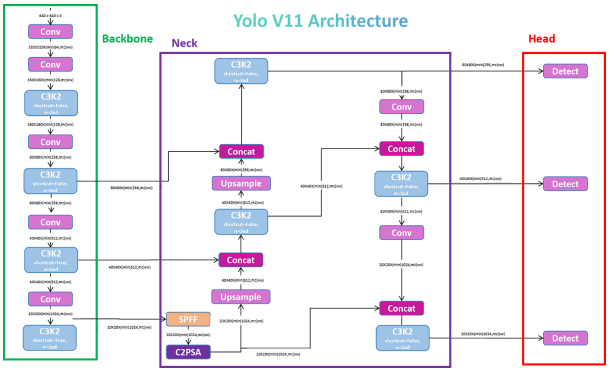

Fig. 2. YOLOv11 model base architecture [5]

The backbone is responsible for extracting hierarchical visual features from the input image of dimension $640 \times 640 \times 3$. It comprises a sequence of convolutional layers, with progressively decreasing spatial dimensions and increasing channel depth. The downsampling occurs through convolutional operations with stride 2 and max-pooling.

The neck of YOLOv11 facilitates the aggregation and upsampling of multi-scale features to support detection at different resolutions. It employs a feature pyramid network structure with:

- Upsample layers to project lower-resolution deep features back to higher resolutions.
- Concat operations that fuse upsampled features with those from earlier stages of the backbone to retain both spatial and semantic information.
- C3K2 blocks, again used here to refine the combined features at each scale before passing them forward.

The detection head is composed of two parallel branches for the two output resolutions (40×40 and 20×20), allowing the model to detect objects of different sizes. Each branch ends with a Detect module, which outputs bounding boxes, objectness scores, and class probabilities. The design allows for precise localization and classification while maintaining high inference speed.

For the project, several versions have been used: nano, small, medium, large and extra-large, with the parameters in Table I.

TABLE I
YOLOv11 MODELS PARAMETERS [2]

Model	mAPval	params (M)
YOLO11n	39.5	2.6
YOLO11s	47.0	9.4
YOLO11m	51.5	20.1
YOLO11l	53.4	25.3
YOLO11x	56.9	56.9

C. RT-DETRv2 Architecture

The RT-DETRv2 (Real-Time DEtection TRansformer) architecture, illustrated in Figure 3, adopts a hybrid encoder-decoder design that combines convolutional backbones with transformer-based modules. Its design aims to maximize detection performance while maintaining real-time inference capabilities, particularly in cluttered and unstructured environments.

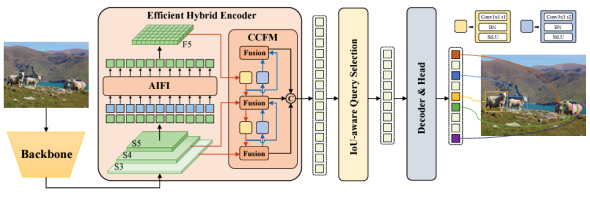

Fig. 3. RT-DETRv2 model base architecture [6]

The processing pipeline begins with a conventional convolutional **backbone**, which extracts multiscale features from the input image. This stage includes layers S3–S5, which correspond to three progressively coarser spatial scales used for hierarchical feature extraction.

These multiscale features are then passed to an **Efficient Hybrid Encoder**, composed of modules for fine-grained reasoning within each scale of features. Following the encoder, the model leverages an **IoU-aware Query Selection** strategy. RT-DETRv2 selects queries based on their predicted Intersection over Union (IoU) alignment with ground-truth boxes. This leads to faster convergence and improved localization, particularly in real-time settings.

The **Decoder and Detection Head** then processes the selected queries. The detection head outputs final bounding boxes and class probabilities.

For the project, the models used were the **RT-DETRl** and **RT-DETRxl** variants, whose parameters are reported in Table II.

TABLE II
RT-DETRV2 MODELS PARAMETERS [2]

Model	mAPval	params (M)
RT-DETRl	53.4	42
RT-DETRx	54.3	76

D. DBSCAN Clustering Method

In the final stage of the pipeline, a post-processing step is introduced to group detected waste objects belonging to the same class and located in spatial proximity. This is accomplished using the DBSCAN algorithm [3]. Unlike centroid-based methods such as K-Means, DBSCAN does not require the number of clusters to be specified a priori.

DBSCAN operates by categorizing each point (in this context, each detected object's bounding box) as either a *core point*, a *border point*, or *noise*, based on the density of points within its neighbourhood. The neighbourhood is defined by a user-specified radius $\varepsilon > 0$, which determines the maximum distance at which two points are considered to be neighbours. In the case of bounding boxes the distance metric is computed as the distance between the boxes.

In the context of waste detection, the ε parameter plays a central role in determining the spatial tolerance for grouping objects. A small ε may result in over-fragmentation, treating objects that are only slightly separated as distinct clusters; conversely, a large ε may incorrectly group distant objects, including elements of other waste categories in the cluster.

Once clusters are formed, the system computes bounding boxes that enclose each cluster. To quantitatively evaluate clustering performance, a cluster-level mean Average Precision at IoU threshold 0.5 (mAP@50) is computed against ground-truth groupings.

IV. DATASET ANALYSIS AND PRE-PROCESSING

A. Dataset Overview

The *YOLO Waste Detection* dataset used in this work is the ProjectVerba collection released on Roboflow, providing **5 460** unique RGB images and **7 896** YOLOv11-formatted bounding boxes across 42 fine-grained waste labels. The dump already contains paired `/images` and `/labels` directories, and no major data cleaning process was required since no orphans images or annotations were present. Like most publicly available waste-detection datasets, the one used in this work exhibits a pronounced class imbalance: the dominant category, *Aluminum can*, accounts for 1,878 bounding boxes, while the rarest classes, *ceramic* and *wood*, appear only twice each. Such severe class skew is common in litter detection benchmarks and is known to degrade the performance of object detectors unless mitigated through augmentation and resampling strategies.

B. Remapping to Waste Categories

Since our objective is to produce an Intelligent System to support an automated waste sorting device, only five macro categories are operationally relevant: `glass`, `metal`, `organic`, `paper`, and `plastic`. All 42 original labels were therefore remapped into these macro classes via a look-up table.

TABLE III
MACRO-CLASS DISTRIBUTION AFTER REMAPPING (*YOLO Waste Detection* V1)

Macro class	Instances	Share (%)
Glass	554	7.0
Metal	2 264	28.7
Organic	760	9.6
Paper	1 278	16.2
Plastic	3 040	38.5
Total	7 896	100

Even after remapping the original 42 labels into the five operational categories, the dataset remains heavily unbalanced: *plastic* still accounts for 38.5 % and *metal* for 28.7 % of all annotations, while *glass*, *paper*, and *organic* are significantly lower. This persistence of a Skewed Class Distribution (SCD) reflects the findings of Sayed *et al.*, who observed a similar dominance of high-consumption materials in TrashNet and addressed it through class-aware resampling and augmentation [16].

Such imbalance typically arises from contextual biases in how waste-detection datasets are collected. For instance, the TACO dataset, comprising 1 500 crowd-sourced photos and 4 784 object instances, was specifically curated for in-the-wild and underwater litter detection [17]. Its images, captured on beaches, forest trails, roadsides, and during scuba dives, naturally emphasize common underwater and roadsides waste: the class `cigarette` alone contributes roughly 1,300 masks, while `plastic_bag_&_wrapper` adds nearly another 1,000—together representing over 40% of the dataset.

This kind of class skew has well-documented impacts on training, with YOLO-style detectors suffering drops of up to 11 mAP points when rare classes fall below a 10% frequency threshold.

C. Cross Validation

To ensure comparability across models, the dataset was initially split into a *training set* (80% of all samples) and a *test set* (20%). The latter was held fixed and used as a common evaluation benchmark for all model variants. Moreover, in order to obtain a statistically relevant performance of the adopted models and an unbiased comparison between the YOLO and RT-DETR families, we adopted a *stratified* k-fold cross-validation process with $k = 5$. At each iteration, four folds (80% of the data) of the train set served for training while the remaining fold (20%) acted as a validation set; the process was then rotated until every fold had appeared as the validation fold exactly once.

D. Data Agumentation (train set only)

Both classical and modern computer vision techniques were applied *exclusively to the training folds* to augment train data for each fold, (remaining fold for testing remained untouched):

a) Classic photometric & geometric transforms: The primary goal of these augmentations is to expand the training distribution and introduce real-world variability such as changes in viewpoint, scale, and illumination while also *oversampling minority classes*. This is expected to make the model more robust in recognizing objects under diverse lighting conditions and perspectives. We therefore adopted the following transformations, each applied with equal probability, to obtain from 1 to 4 transformed variants for each input image:

- horizontal or vertical flip;
- rotation in {90°, -90°, 180°};
- exposure shift in the range $\pm 50\,\%$.

b) Synthetic Image Generation: To completely resolve the class imbalance in our dataset, discussed in Section IV, we adopted a *copy–paste + render* pipeline inspired by Ghiasi *et al.* [20]. Unlike typical augmentation strategies that focus solely on equalizing class frequencies, our approach was designed to *both* re-balance the training set while *preserving its statistical properties*, such as number of objects per-image and spatial layout. In particular we computed the following metrics over the training set:

- **Average number of objects per image:** $\mu = 1.45$;
- **Centroid distribution:** spatial heatmaps show a concentration of bounding boxes centers near the image center (see Figure 4);
- **Mean bounding-box area:** approximately 40 % of the image area;
- **Mean bounding-box aspect ratio:** $\lambda = 0.85$.

The synthetic image generation process was structured in three main steps:

1) **Object Bank Construction:** Pixel-accurate object crops were extracted from segmentation masks in the TACO dataset, and each instance was remapped to one of the five waste categories (*glass*, *metal*, *organic*, *paper*, *plastic*).
2) **Image Synthesis:** Two complementary strategies were employed:

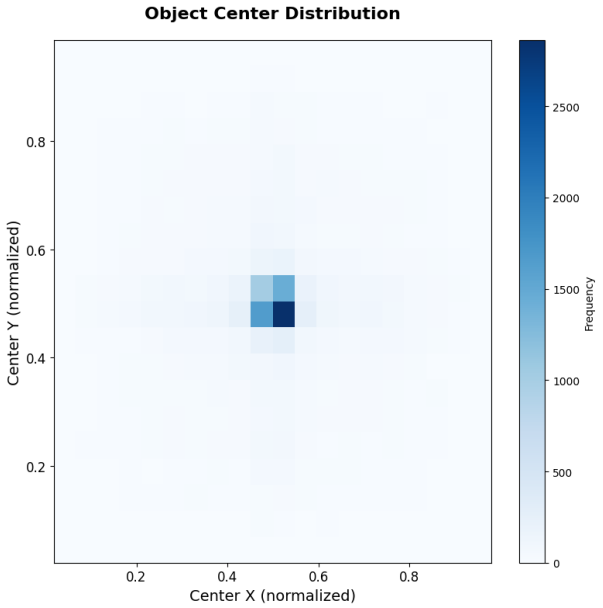

Fig. 4. Spatial heatmap of the distribution of the centers of objects' bounding boxes

   a) **Augmented Images:** In 70% of the cases, synthetic objects were pasted onto real backgrounds sampled from the current fold's training set—preserving environmental realism, an example of augmented images can be seen in Figure 5.
   b) **Synthetic Images:** In the remaining 30 %, full synthetic scenes were rendered from scratch using generated backgrounds with randomized lighting, textures, and blur to maximize diversity, an example of synthetic image can be seen in Figure 6 .
3) **Fold-Aware Class Balancing:** For each fold, generation proceeded until the rarest macro-category reached numerical parity with the most frequent one. This yielded an average increase of ∼30% in training data volume and resulted in a near-uniform class distribution.

The combined augmentation pipeline, which included both traditional transformations and synthetic generation, expanded the training set to approximately **27,000 images**. However, training large models—such as YOLOv11 in its large and extra-large configurations—on the full dataset would have been computationally prohibitive. To address this, after several tests, we applied a random subsampling strategy to reduce the training set to **10 000 images**, while preserving the class distribution uniformity achieved through augmentation. This number of images emerged as a reasonable compromise between avoiding the risk of model overfitting and retaining sufficient dataset diversity, especially considering that synthetically generated images do not carry the same amount of information as real ones [20].

E. Summary

Through class remapping, stratified 5-fold cross-validation, and a two-stage augmentation strategy (classical + synthetic),

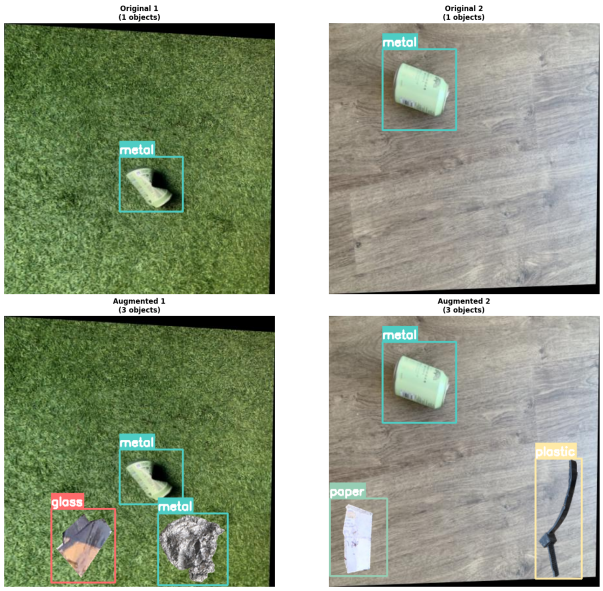

Fig. 5. Example of augmented images, on the top two images from the dataset, on the bottom the respective augmented images

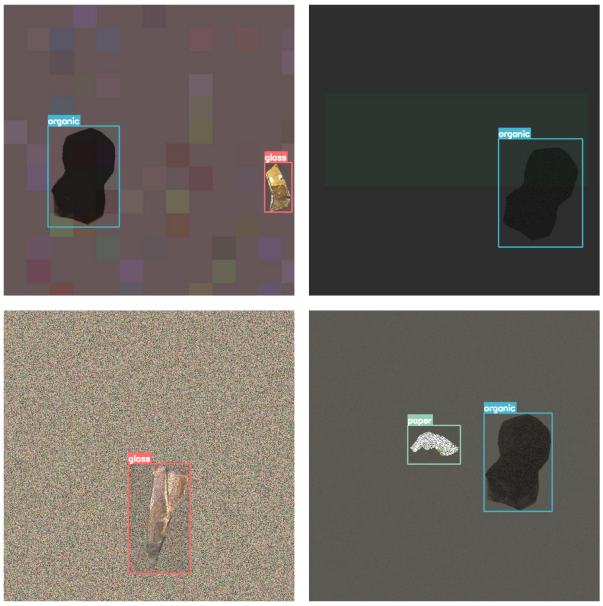

Fig. 6. Example of 4 synthetic images in a 2x2 square

we constructed balanced training sets of nearly **10 000 images per fold**. This augmentation pipeline not only increases dataset cardinality and balances classes distribution, but also preserves key statistical properties of the original data, ensuring consistency and enhancing generalization.

V. Experiments and Results

A. Impact of Augmentation on Classification Performance

To assess the effectiveness of our data augmentation strategy, we compared the model's performance before and after applying the augmentation pipeline (classical + synthetic). The primary metric considered is mAP@50, which shows a notable improvement—from approximately **0.76** to **0.80** before and after augmentation respectively. Other metrics adopted are Precision, Recall and F1-Score to measure respectively the quality of predictions, the presence of undetected objects and a balance of the two values.

Figure 7 shows an example of normalized confusion matrix of the model trained without data augmentation, whereas Figure 8 shows and example of normalized confusion matrix of the model trained with augmented train set.

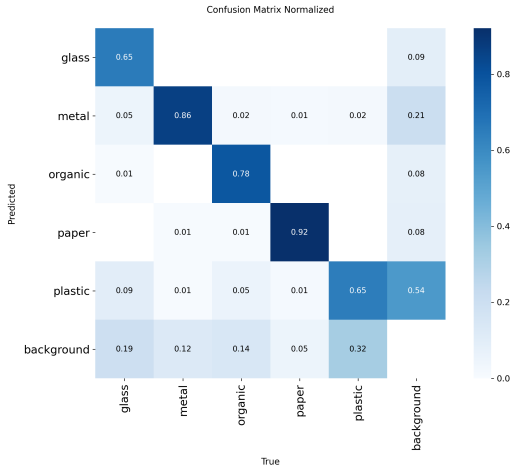

Fig. 7. Normalized confusion matrix before applying the augmentation pipeline.

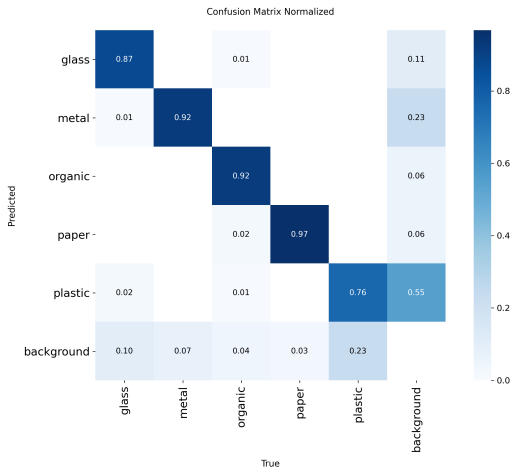

Fig. 8. Normalized confusion matrix after applying the augmentation pipeline.

The changes are particularly evident for underrepresented classes. For example, **glass**, which accounts for only 12 % of the dataset, the fraction of correct prediction rises from 0.65 to **0.87**. Similarly, **organic** improves from 0.78 to **0.92**. More frequent classes also benefit slightly: **plastic**—the dominant category at 38.6%—sees its accuracy increase from 0.65 to **0.76**.

TABLE IV
PER-CLASS AP@50 FOR EACH MODEL VARIANT. RESULTS ARE AVERAGED OVER MULTIPLE FOLDS. MODELS MARKED WITH * WERE EVALUATED ONLY ON ONE FOLD.

Model	Family	Size	Glass	Metal	Organic	Paper	Plastic	mAP@50
YOLOv11-n	YOLOv11	n	0.87	0.90	0.70	0.93	0.61	**0.804**
YOLOv11-s	YOLOv11	s	0.89	0.91	0.73	0.94	0.62	**0.818**
YOLOv11-m	YOLOv11	m	0.89	0.91	0.74	0.93	0.64	**0.825**
YOLOv11-l*	YOLOv11	l	0.84	0.92	0.72	0.94	0.61	**0.809**
YOLOv11-x*	YOLOv11	x	0.86	0.91	0.73	0.92	0.65	**0.811**
RT-DETRv2-l*	RT-DETRv2	l	0.83	0.86	0.65	0.91	0.51	**0.754**
RT-DETRv2-x*	RT-DETRv2	x	0.85	0.85	0.65	0.87	0.48	**0.740**

Although plastic is the most represented class in the dataset, its improvement in accuracy can be due to the increased object variability introduced through data augmentation. By expanding the training sets with noisier images, varied lighting conditions, and multiple viewpoints, the model was able to correctly classify more plastic objects. This is particularly beneficial given the high diversity plastic objects, which includes different shapes, colors, and sizes. As a result, the model became more robust and accurate in correctly detecting objects.

This comparison was performed using the **YOLOv11n** model with base train set and augmented train set. Both models were trained for **100 epochs**, with input image size of **320×320 pixels**. Given the significant improvements, observed both in the mAP@50 metric and in the per-class confusion matrix, all following trainings, including those involving larger YOLOv11 variants and models from the RT-DETR family, were conducted using the augmented dataset.

B. YOLO and RT-DETR performance evaluation

a) Object Detection: As mentioned earlier, the evaluation process was based on a **5-fold cross validation** to ensure statistical robustness and mitigate bias due to data partitioning. The reported performance corresponds to the average **mAP@50** computed across all five folds on the fixed test set, discussed in Section IV, to ensure comparable results across the different models.

However, for larger model variants, specifically the **Large** and **X-Large** versions from both the **YOLO** and **RT-DETR** families, the computational cost of training was significantly higher. As a result, evaluation for these models was limited to the **first fold only**. While this reduces the statistical confidence for these specific results, it still provides a meaningful comparison.

Table IV and Table V summarize the per-class and global performance of each model, following the evaluation protocol previously described.

From the same tables, it is evident that increasing model capacity does not result in a proportional improvement in accuracy. On one side, moving from the compact versions of **YOLOv11** (sizes *n* and *s*) to the medium variant (*m*) yields only a modest gain in *mAP@50* ($0.804 \rightarrow 0.825$, $\Delta \approx +0.02$). On the other side, the largest models—both from the YOLO and RT-DETR families—show either negligible improvements or, in some cases, slightly worse performance compared to their smaller counterparts. For instance, **RT-DETRv2–x** underperforms **YOLOv11–m**, achieving a lower *mAP@50* ($0.740 \rightarrow 0.825$, $\Delta \approx -0.085$) despite having substantially more parameters and computational cost.

TABLE V
PRECISION (P), RECALL (R), AND F1 SCORES FOR EACH MODEL VARIANT.

Model	P	R	F1
YOLOv11-n	0.829	0.761	0.794
YOLOv11-s	0.839	0.766	0.801
YOLOv11-m	0.839	0.773	0.804
YOLOv11-l	0.831	0.758	0.793
YOLOv11-x	0.840	0.749	0.792
RT-DETRv2-l	0.829	0.699	0.758
RT-DETRv2-x	0.799	0.711	0.753

This behavior can be attributed to the tendency of larger models to overfit the training data, which compromises their ability to generalize to unseen examples. Supporting this observation, for all large and extra-large models, across both the **YOLOv11** and **RT-DETRv2** families, the final training *mAP@50* exceeds **95%** across all folds, indicating overfitting. For the same reason, increasing the model size in YOLO the precision increases while the recall decreases, leading to an overall performance loss: the model recognizes correctly the object that detects but tends to generate false negatives, probably due to inability of generalization.

As mentioned, the most performative model is **YOLOv11-m**, for which we report the confusion matrix in Figure 9, the P-R graph and the F1-confidence graph in Figure 10:

The figures show very good detection and classification performances on glass, metal and paper, while worse predictions on organic and plastic, probably due to their complexity and variety. The F1 score reveals that detections can be both precise and effective even with high confidence thresholds.

b) Real-Time Performance: From a performance standpoint, increasing model size results in a substantial drop in inference speed, measured in frames per second (FPS). For example, **YOLOv11–n** achieves 31.6 FPS on average, while the largest variant, **YOLOv11–x**, drops to just 8.3 FPS—a reduction of approximately 74%. This pattern reflects the well-known trade-off: although larger models may offer higher

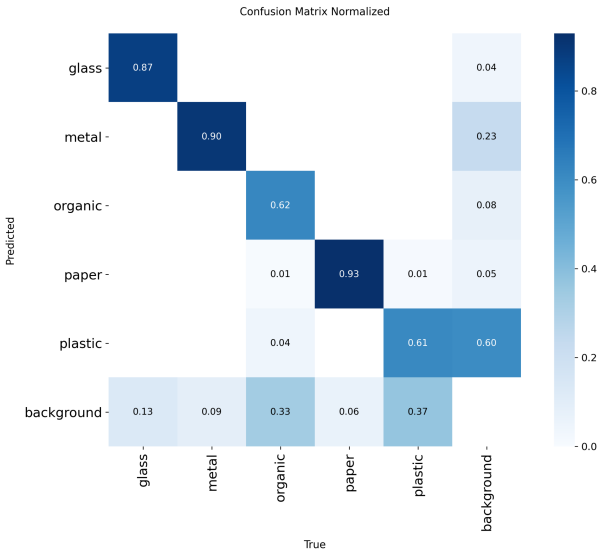

Fig. 9. Confusion matrix of YOLO11-m model on the test set.

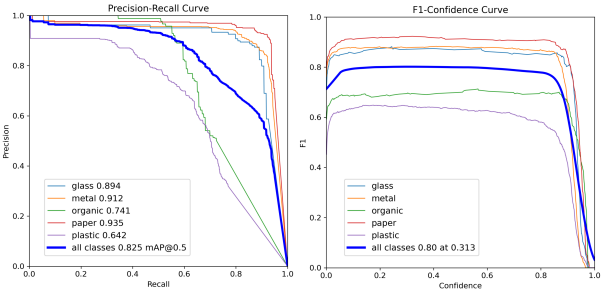

Fig. 10. Precision-Recall graph and F1-Confidence graph for YOLOv11-m model on the test set.

mAP@50, they demand significantly more computation, which affects real-time capability of the system.

Table VI summarizes the inference speed across all evaluated models. All tests were conducted on an **NVIDIA Tesla T4** GPU with 16 GiB of VRAM.

TABLE VI
AVERAGE INFERENCE SPEED AND STANDARD DEVIATION IN FPS FOR EACH MODEL VARIANT.

Model	Family	Size	FPS
YOLOv11–n	YOLOv11	n	31.6
YOLOv11–s	YOLOv11	s	23.4
YOLOv11–m	YOLOv11	m	15.4
YOLOv11–l	YOLOv11	l	14.0
YOLOv11–x	YOLOv11	x	8.3
RT-DETRv2–l	RT-DETRv2	l	13.4
RT-DETRv2–x	RT-DETRv2	x	5.1

As the table shows, RT-DETR models are consistently slower than their YOLOv11 counterparts. In particular, **RT-DETRv2–x** reaches only 5.1 FPS—about 39% slower than **YOLOv11–x**. This drop in speed is primarily due to the quadratic complexity of the attention mechanism in RT-DETR's transformer decoder, as opposed to the fully-convolutional pipeline of YOLOv11.

C. Clustering Metrics

To quantitatively assess the quality of the cluster predictions, we introduce a specific evaluation protocol that extends the traditional object detection metrics (e.g., mAP@50) to the cluster level. The proposed methodology is based on the insight that, in real-world scenarios involving a large number of objects of the same category (e.g., multiple plastic bottles), the exact detection of every individual bounding box is less important than the system's ability to identify object groupings. For this reason, clustering evaluation is performed in two parallel phases: one based on ground-truth labels and one on the predicted bounding boxes.

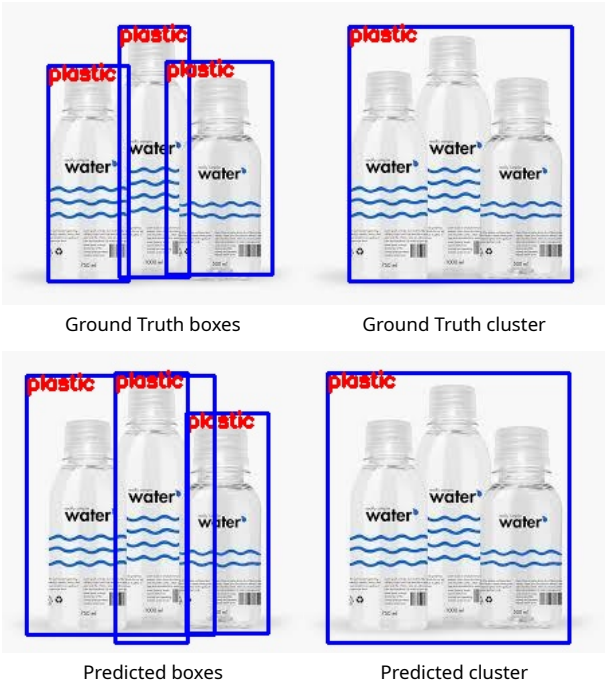

Fig. 11. Comparison between clusters derived from ground truth and predicted bounding boxes

The process begins by applying the DBSCAN algorithm independently to the ground-truth bounding boxes and to the predicted detections. This two separate sets of clusters per image serve one as reference, the other as the model's output. Given centers coordinates, width and height of each box, the distance metric is computed as

$$dx = max(0, |x1_c - x2_c| - (w1 + w2)/2)$$

$$dy = max(0, |y1_c - y2_c| - (h1 + h2)/2)$$

$$d = \sqrt{dx^2 + dy^2}$$

The metric used to evaluate clustering performance is a cluster-level mean Average Precision at 0.5 Intersection over Union (**cluster mAP@50**). A predicted cluster is considered

Fig. 12. Effect of the ε parameter on the clustering process to avoid clusters overlaps. On the left the original bounding boxes, on the right the clusters generated with DBSCAN and $\varepsilon = 20$px.

a true positive if it overlaps (IoU > 0.5) with a corresponding ground-truth cluster of the same category.

An important design choice lies in the selection of the ε parameter. It is chosen such that clusters do not merge inappropriately across category boundaries: for example, if a metal bar is spatially located between two groups of plastic elements, the DBSCAN configuration prevents it from being included in an overlapping plastic cluster (Figure 12).

The main benefit of this evaluation strategy lies in its robustness to discrepancies in object-level predictions. For instance, if the model predicts two adjacent plastic bottles as a single bounding box, while the ground truth contains them as two separate boxes, the resulting cluster may still be correct. This approach allows the system to be tolerant to over- or under-segmentation as long as the overall spatial and categorical grouping is preserved.

TABLE VII
MAP@50 AND CLUSTER MAP@50 (C_MAP@50) FOR EACH MODEL VARIANT.

Model	mAP@50	c_mAP@50
YOLOv11-n	0.804	**0.8585**
YOLOv11-s	0.818	**0.8544**
YOLOv11-m	0.825	**0.859**
YOLOv11-l	0.809	**0.856**
YOLOv11-x	0.811	**0.855**
RT-DETRv2-l	0.754	**0.827**
RT-DETRv2-x	0.740	**0.815**

In fact, looking at Table VII it is possible to observe that the value of the mAP@50 computed on clusters is higher than that computed for per-object prediction. The value of c_mAP@50 can be considered the most effective metric to represent the performances of the model and the baseline for further enhancements.

VI. Conclusion

This study presented *Smart Recycling*, an end-to-end pipeline for fold-aware data augmentation, both with classical computer vision techniques and synthetic generation, with a real-time waste detector, and a density-based post-processor to support automated waste sorting and optimal grasping scheduling. Unlike prior approaches that focus solely on per-object accuracy, our work integrates (i) a two-stage augmentation strategy that prevents classes unbalance while preserving the original statistical properties of the YOLO WASTE DETECTION dataset, (ii) a stratified five-fold cross-validation protocol that produces statistically relevant results, and (iii) a DBSCAN clustering module that converts raw bounding boxes into graspable groups suitable for single-pick manipulation.

More specifically, the copy–paste + synthetic augmentation pipeline raised the rarest class (glass) from 7 % to a uniform distribution among all the different waste classes, lifting the YOLOv11-n baseline from 0.76 to 0.80 mAP@50.

The best performance in terms of detection accuracy was achieved by YOLOv11–m, which reached a mean mAP@50 of 0.83. RT-DETR variants, particularly RT-DETR–l and RT-DETR–x, showed competitive accuracy (mAP@50 of 0.75 and 0.74 respectively), but were affected by signs of overfitting—likely due to their larger capacity relative to the training set size. Moreover, while RT-DETR models exhibited strong localization precision, their inference speed was significantly lower. RT-DETR–l ran at 13.4 FPS and RT-DETR–x at only 5.1 FPS, well below real-time thresholds on the tested hardware. This performance bottleneck is attributed to the quadratic complexity of the attention mechanism in transformer-based backbones, which makes them unsuitable for latency-constrained applications in this context.

Conversely, YOLOv11 variants demonstrated an excellent balance between speed and accuracy. YOLOv11–n and YOLOv11–s reached 31.6 FPS and 23.4 FPS respectively, with consistent performance across all folds. These models are therefore more appropriate for real-time deployment in resource-limited environments, such as smart bins or mobile robotic systems.

Cluster-level evaluation with DBSCAN further validated the effectiveness of the method, converting per-box gains into high-quality group predictions suitable for robotic grasping. The experiments shows that applying a clustering algorithm the value of the mAP@50 metrics can increase between +3 and +7. This confirms that lightweight detectors, when properly trained and post-processed, are sufficient to meet both the precision and throughput requirements of autonomous recycling systems.

References

[1] **F. Barcherini, A. Di Matteo, "AI waste detection", repository of the project, 2025. [Online]. Available: https://github.com/Francesco-Barcherini/AI-waste-detection**

[2] Ultralytics, "Ultralytics Yolo and RT-DETR" 2025. [Online]. Available: https://github.com/ultralytics/ultralytics

[3] Scikit Learn, "DBSCAN", 2025. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

[4] Scikit Learn, "pairwise_distances", 2025. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise_distances.html#sklearn.metrics.pairwise_distances

[5] S. Nikhileswara Rao, "YOLOv11 Architecture Explained: Next-Level Object Detection with Enhanced Speed and Accuracy", 2024. [Online]. Available: https://medium.com/@nikhil-rao-20/yolov11-explained-next-level-object-detection-with-enhanced-speed-and-accuracy-2dbe2d376f71

[6] W. Lv, Y. Zhao, Q. Chang, K. Huang, G. Wang, and Y. Liu, "RT-DETRv2: Improved Baseline with Bag-of-Freebies for Real-Time Detection Transformer," *arXiv preprint arXiv:2407.17140*, 2024. [Online]. Available: https://arxiv.org/abs/2407.17140

[7] Esri, "RT-DETRv2 Object Detector", 2025. [Online]. Available: https://developers.arcgis.com/python/latest/guide/rt-detrv2-object-detector

[8] F. Fotovvatikhah, I. Ahmedy, R. Md Noor, and M. U. Munir, "Artificial intelligence in smart waste management: A systematic review," *Sensors*, vol. 25, art. no. 3181, 2025.

[9] H. Umer, A. Ahmed, F. Ali, S. S. Ali, and M. A. Khan, "A systematic literature review on smart waste management using machine learning," in *Proc. Int. Conf. Comput. (MAJICC)*, Islamabad, Pakistan, 2022, pp. 1–6.

[10] L. Zhao, Y. Pan, S. Wang, Z. Abbas, M. S. Islam, and S. Yin, "Skip-YOLO: Domestic garbage detection using deep learning method in complex multi-scenes," *Int. J. Comput. Intell. Syst.*, vol. 16, art. no. 139, 2023.

[11] A. Arishi, "Real-time household waste detection and classification for sustainable recycling: A deep learning approach," *Sustainability*, vol. 17, no. 5, art. no. 1902, 2025.

[12] P. A. Rajeev, V. Dharewa, D. Lakshmi, V. G. Vishnuvarthanan, J. Giri, T. Sathish, and M. Alrashoud, "Advancing e-waste classification with customizable YOLO-based deep learning models," *Sci. Rep.*, vol. 15, art. no. 18151, 2025.

[13] J. Son and Y. Ahn, "AI-based plastic waste sorting method utilizing object detection models for enhanced classification," *Waste Manag.*, vol. 193, pp. 273–282, 2025.

[14] M. M. Abo-Zahhad and M. Abo-Zahhad, "Real-time intelligent garbage monitoring and efficient collection using YOLOv8 and YOLOv5 deep learning models for environmental sustainability," *Sci. Rep.*, vol. 15, art. no. 16024, 2025.

[15] I. H. Al Amin, D. V. Setyadarma, and S. Wibisono, "Integration of the Faster R-CNN algorithm for waste detection in an Android application," *Revue d'Intell. Artif.*, vol. 37, no. 6, pp. 1407–1414, 2023.

[16] G. I. Sayed, M. A. Elfattah, A. Darwish, and A. E. Hassanien, "Intelligent and sustainable waste classification model based on multi-objective beluga whale optimization and deep learning," *Environ. Sci. Pollut. Res.*, vol. 31, no. 21, pp. 31492–31510, Apr. 2024.

[17] P. F. Proença and P. Simões, "TACO: Trash Annotations in Context for Litter Detection," *arXiv preprint* arXiv:2003.06975, 2020.

[18] H. Panwar, P. K. Gupta, M. K. Siddiqui *et al.*, "AquaVision: Automating the detection of waste in water bodies using deep transfer learning," *Case Stud. Chem. Environ. Eng.*, vol. 2, art. 100026, 2020.

[19] M. Valdenegro-Toro, D. C. Padmanabhan, D. Singh, B. Wehbe, and Y. Petillot, "The Marine Debris Forward-Looking Sonar Datasets," *arXiv preprint* arXiv:2503.22880, 2025.

[20] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation," in *Proc. CVPR*, 2021, pp. 2918–2928.

[21] S. Zhang, N. Wang, and H. Li, "Distribution-aware augmentation for long-tailed object detection," *IEEE Trans. Image Process.*, vol. 33, pp. 821–835, 2024.