COmmunity-Based Organized Littering

PRIN: PROGETTI DI RICERCA DI RILEVANTE INTERESSE NAZIONALE

Bando 2022 PNRR

Deliverable D1.2

A Lightweight model for waste recognition, volume estimation, and material segmentation, together with the labeled dataset collected by the community

https://prin-cobol.github.io

Funded by the European Union
NextGenerationEU

Project Number	:	P20224K9EK
Project Title	:	COBOL: COmmunity-Based Organized Littering

Deliverable Number	:	D1.2
Title of Deliverable	:	A Lightweight model for waste recognition, volume estimation, and material segmentation, together with the labeled dataset collected by the community
Nature of Deliverable	:	Report / Other
Dissemination level	:	Public
License	:	Creative Commons Attribution 3.0 License
Version	:	1.0
Contractual Delivery Date	:	M12
Contributing Objective/Task	:	O1
Editor(s)	:	Leonardo Mariani (UNIMIB)
Author(s)	:	Luciano Baresi (PoliMI), Simone Bianco (UniMIB), Amleto Di Salle (GSSI), Ludovico Iovino (GSSI), Daniela Micucci (UniMIB), Leonardo Mariani (UniMIB), Luciana Brasil Rebelo dos Santos (GSSI), Maria Teresa Rossi (UniMIB), Raimondo Schettini (UniMIB)

Glossary, acronyms & abbreviations

Item	Description
COBOL	COmmunity-Based Organized Littering
CRUD	Create, Read, Update, Delete

Table of Contents

1 Introduction..... 4
2 The lightweight model for waste recognition..... 5
3 The collected dataset..... 6
4 Results..... 7
5 Conclusions and remarks..... 7
6 References..... 8

1 Introduction

Littering is a growing problem that afflicts many cities and communities around the world. The improper disposal of waste contributes to pollution, harms wildlife, and degrades natural landscapes.

Automatic litter detection, thanks to its ability to identify waste in various environments quickly and accurately, can provide continuous monitoring, cover larger areas, and reduce the reliance on human resources. These advantages make automatic litter detection an essential component of modern waste management strategies.

One of the most promising approaches to achieving comprehensive and detailed surveys for litter detection is through the collective efforts of citizen science. Citizen science involves the participation of volunteers from the general public in scientific research activities, leveraging their collective power to gather data over vast areas and time periods. With the widespread availability of smartphones, citizen scientists can now use their devices to capture images and report instances of litter, providing valuable data for environmental monitoring.

We started developing the litter detector using a dataset in state of the art, and in the meantime acquiring our own dataset.

Among the different existing datasets, the TACO (Trash Annotations in Context) dataset is one of the best ones publicly available for waste detection, as it contains realistic scenarios with a wide variety of waste thus permitting the training of litter detection models able to operate on images in the wild, i.e. with uncontrolled acquisition conditions.

Good performance has been reported on the TACO dataset, but they are achieved by large models as for example YOLO-v5x with a model size of more than 170 MB, making difficult its deployment on edge devices with limited computational resources as for example low- and mid-range smartphones.

In this deliverable we propose to tackle the automatic litter detection problem using the lightest object detection models currently available in the state of the art: YOLO-v5 and YOLO-v8 considering only the tiny and small variants. The challenge is to train these models trying to obtain the best possible performance on the TACO dataset, and compare them with the results in the state of the art.

The trained models are then compressed with different quantization levels, as for example half precision FP16 and INT8 quantization to investigate the trade-off between model size and detection performance.

On the same dataset we also performed the task of material segmentation, while for volume estimation a new dataset named UniMiB Trash dataset has been collected and carefully annotated.

2 The lightweight model for waste recognition

In this work we experiment with the YOLO object detector. The name YOLO, which stands for “You Only Look Once”, is a state-of-the-art, real-time object detection algorithm, introduced in 2015 [Redmon2016look]. YOLO belongs to the category of one-stage detectors [Zou2023object] and spatially separates bounding boxes and associates probabilities to each of the detected objects using a single pass over the input image with a Convolutional Neural Network (CNN).

In this work we consider its most popular versions, implemented in the Ultralytics library, i.e., YOLO-v5 [yolov5_ultralytics] and YOLO-v8 [yolov8_ultralytics], focusing in particular on the *tiny* and *small* models. Both YOLO-v5 and YOLO-v8 use input images of size 640x640, while a variant of YOLO-v5, available in both tiny (YOLO-v5n6u) and small (YOLO-v5s6u) model sizes, uses input images of size 1280x1280.

All the methods are trained with the same default hyperparameters for a total of 100 epochs, with automatic batch size selection, and automatic optimizer selection. Three additional image augmentations are added to the default ones:

- *flipud* that flips the image upside down with the specified 0.5 probability
- *degrees* that rotates the image randomly within the specified [-10, 10] degrees range
- *copy_paste* that copies objects from one image and pastes them onto another, resulting particularly useful for increasing object instances and learning object occlusion.

Once the training of a model is complete, as a further optimization we tune the confidence threshold *conf*, which is responsible for discarding the detections having an associated confidence score lower than its value.

In order to qualitatively evaluate the effect of this tuning, we report in the following figure the detections returned by YOLO-v5s on a couple of TACO images with the default confidence (i.e., *conf*=0.001) and with the confidence set at 0.5.

From the examples reported it is possible to notice how the default confidence value tends to increase the number of false positives, while the increased confidence maintains only the most precise detections.

Concerning the confidence threshold tuning, the optimization of its value is carried out on the validation set by choosing the threshold value in the set 0.001, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95 the one resulting in the highest mAP50.

3 The collected dataset

The need to collect a dataset from scratch comes due to several reasons, which don't allow to build a strong Deep Learning model to detect litter in the wild, and especially trying to create an object detection that would be effective in a multi class problem.

We report below the reasons for collecting a new dataset:

- Insufficient images in existing datasets: The number of images in the considered datasets is quite limited for Deep Learning models: TACO and PlastOPol contain only 1,500 and 2418 images respectively. Combining the datasets would result in a total of 3,918 images, which is still not a substantial amount.
- Realistic scenarios only: in existing datasets, some images do not actually depict litter in real-world conditions. While the objects themselves could potentially be the same, the complexity of real-world scenarios is entirely absent—factors such as possible occlusions (plants, leaves, other waste), deterioration, lighting conditions, and background variability are excluded. Given that the number of images is already limited, the presence of unrealistic examples further reduces the usefulness of the datasets.
- Prevalence of single or limited waste scenarios: The complexity of TACO, lies in the fact that the objects to be detected are often very small. Although this realism can be beneficial — and sometimes even excessive — it is not sufficient on its own to fully represent the complexity of the real world. Typically, only one or a few pieces of litter are present in a single image, whereas many real-world reports may involve piles of waste resembling illegal dumps.

The following is a detailed description of all the steps involved in the dataset collection process. A free tool from Roboflow's platform was used to support each step.

Collecting Images

Collecting images is the first and fundamental step in building the dataset. Being a human task, it is time-consuming and costly. The images should come from various sources, such as low-end and high-end smartphones, to make the dataset more heterogeneous. Additionally, they should be captured by different people in different places to ensure a variety of framing and scenarios, enabling Deep Learning models to be trained and tested in a real context.

The dataset is collected by ten people, each with their own devices. The majority of the photos were taken in the Milan hinterland, encompassing both urban and rural settings, from the countryside to rivers.

Data Quality Control

The second task is also manual, as the images must be carefully reviewed to ensure they contain truly abandoned waste in nature. Images featuring waste that is not truly abandoned, or those that capture the same litter from multiple angles, should be excluded.

Regarding qualitative factors such as blurring, resolution or lighting conditions, a certain degree of flexibility is needed to create a realistic and heterogeneous dataset, provided the images are not excessively compromised.

This process is critical to the dataset's success, demanding careful time and attention to prevent the inclusion of duplicate, unsuitable or compromised images.

Annotation Process

After collecting the images, a hybrid task is performed using the annotation tool available on the Roboflow platform. A human annotator decides what and how to annotate, using the Segment Anything Model (SAM) to support and improve the segmentation process of the litter. The decision is made to annotate the images with polygons rather than bounding boxes, as this allows the dataset to be used by both segmentation and detection models. While polygons can be easily converted into bounding boxes, the reverse is not possible.

In this annotation process, each instance is also assigned a class label. Four class types are defined: micro, small, medium, and large. The classification is based on the physical size of the waste rather than its type. For the deep learning model, this presents a challenge as it must recognize the class of the waste based solely on its size, independent of the image scale.

The *small* class includes objects like common plastic, glass bottles and cigarette packs. The *medium* class is assigned to items such as shopping bags or normal garbage bags. The *large* class includes objects like large garbage bags, as well as various large objects (e.g., metal sheets, iron objects, mattresses, etc.). The *micro* class includes very tiny objects, which are not considered essential for achieving the main goal. Cigarettes are the most common examples, along with small plastic items, candy wrappers, and others. Detecting these tiny objects would require larger (and slower) Deep Learning models, but this is not a critical goal; detecting objects of significant size is more important than merely identifying cigarettes. However, this class is included in the dataset to evaluate the performance of the current state-of-the-art models and to allow future use when hopefully better models can detect these smaller items.

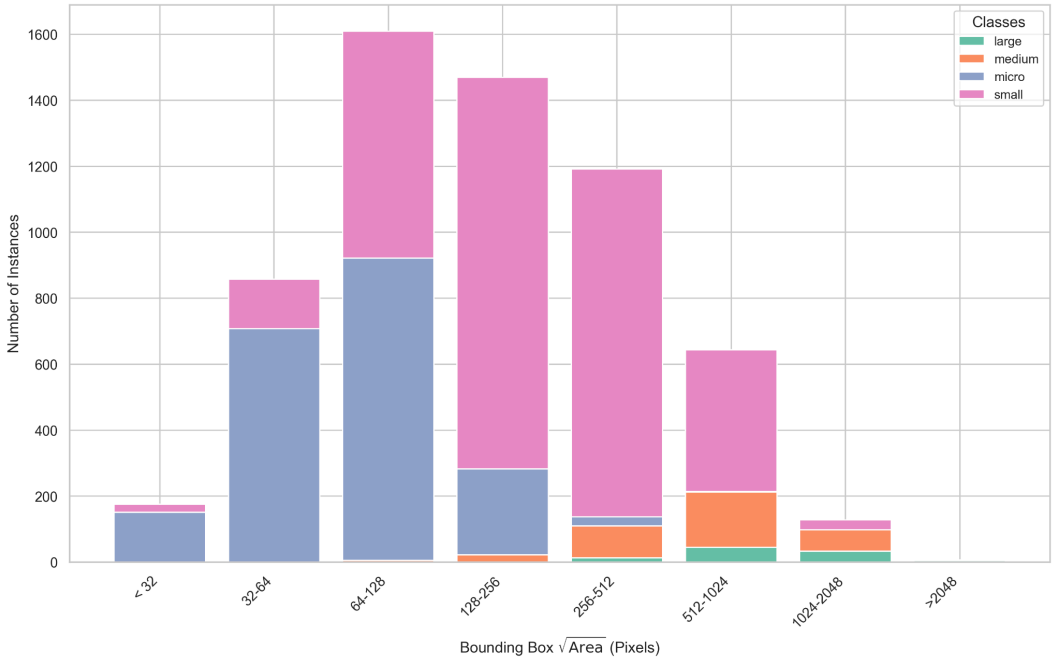
Dataset obtained

The UniMiB Trash dataset comprises 1,232 annotated high resolution images. UniMiB Trash includes a total of 6,289 instances, with an average of about 5.1 annotations per image. These annotations are available in YOLO format as polygons, which facilitates both segmentation and detection tasks. Compared to the TACO and PlastOPol datasets, the number of annotations per image in the UniMiB dataset is significantly higher. The litter instances are segmented and labeled using four different classes, which are categorized not by the type of litter but rather by size. This approach avoids the challenges associated with having either too many poorly represented classes or too few classes with excessive intra-class heterogeneity.

The distribution of images across the classes in the UniMiB Trash dataset is unbalanced, as illustrated in the following figure.

This imbalance reflects the realities of image acquisition, and one of the primary objectives is to address and expand upon this reality. An object's classification is invariant to its scale; for instance, a sofa is classified as *large* regardless of whether it occupies less than 100 pixels or dominates the entire image.

Similarly, a cigarette will always be categorized as *micro*, regardless of whether it appears prominently or minimally within the frame. The following figure presents the size of the bounding boxes in the dataset, categorized by class.



The class *micro* is added for completeness and correctness, as this type of waste is unfortunately very present and potentially useful outside of this task. However, for the ultimate goals of this project, detecting an abandoned sofa or rubbish bags is definitely more significant than detecting a candy wrapper.

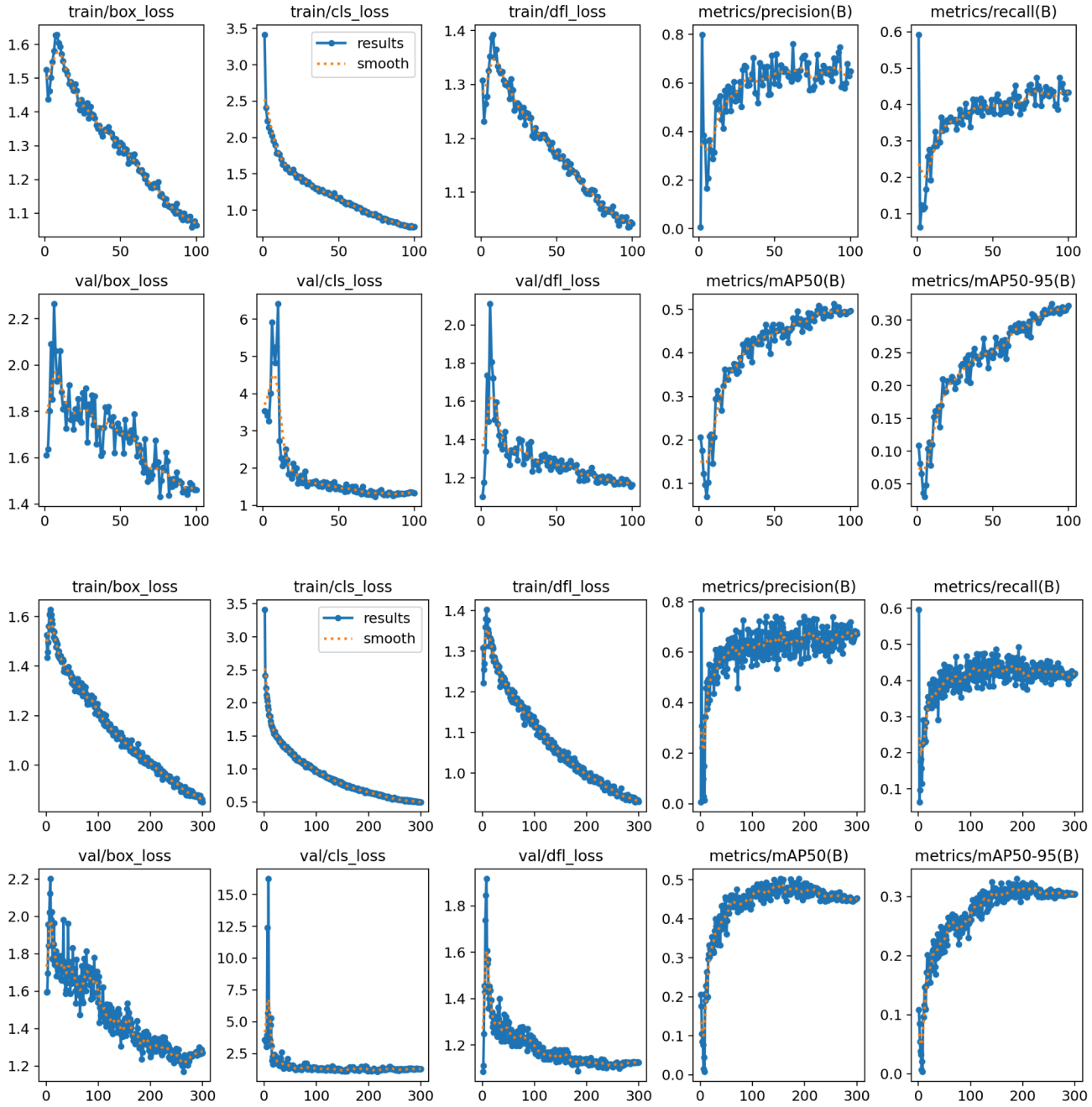
The UniMiB Trash dataset is currently the only dataset that offers a classification of waste instances based on size. This information, which can be relatively easier to capture than the type of waste, could be of great help in the final task, as it could provide useful information for prioritising the waste to be disposed of.

4 Results

Waste recognition on TACO dataset

In this project, only the most popular versions of YOLO implement implemented by Ultralytics library are considered, i.e., YOLO-v5 [yolov5_ultralytics] and YOLO-v8 [yolov8_ultralytics], focusing on the *tiny* and *small* models. The dataset used for waste recognition is TACO-1, i.e. the TACO version where only two classes are available: litter and no litter.

The training trend is illustrated in the following figure, which shows two examples of YOLO v8 nano training, respectively for a total of 100 and 300 epochs.



A total of 100 epochs is sufficient to achieve maximum performance with all models. As the epochs increase, the training trend of YOLO appears to flatten, reaching similar performance levels to those achieved in fewer epochs. Regarding the validation set, a noticeable drop in

the slope of all curves can be observed in the second half of each validation plot. This means that the model has achieved its best possible performance on the validation set. Continuing training would likely lead to the phenomenon of overfitting, where performance improves on the training set while remaining about the same on the validation set.

Tuning of the hyperparameters *conf* and *IoU* as in [Bianco2024efficient], is performed on the validation set. Then, the confidence and IoU of the best evaluated results are used as parameters on the test set. The goal is to develop recall-oriented models rather than precision-oriented ones, placing greater emphasis on recall while still trying to balance performance as best as possible.

The confidence parameter primarily enhances precision, and at lower values, it does not significantly impact recall. As a result, the mAP50 and mAP50-95 can increase by more than 10 percentage points. Tuning the IoU parameter is particularly useful for balancing performance in terms of recall and precision; typically, it allows for an increase in the confidence parameter without compromising recall or even improving it.

The results of the several models on the test set are presented in the following table.

	Default thresholds				Tuned thresholds				Tuned Value		Size
Method	mAP50	mAP50-95	recall	precision	mAP50	mAP50-95	recall	precision	Conf	Iou	(MB)
RetinaNet [7]	73.3	47.2									297.0
Faster R-CNN [7]	75.3	49.6									580.0
Mask R-CNN [7]	73.3	50.2									491.0
EfficientDet-D0 [7]	65.0	51.1									17.0
EfficientDet-D5 [7]	73.2	69.1									136.0
YOLO-v5s [7]	79.9	62.4									15.0
YOLO-v5x [7]	**84.9**	71.1									171.0
YOLO-v5n (this work)	66.6	49.1	57.7	78.6	70.5	$56.3^{+7.2\%}$	60.5	74.3	0.4	0.2	**5.0**
YOLO-v5n6u (this work)	67.9	51.5	61.5	77.0	70.5	57.2	61.4	**77.7**	0.4	0.5	8.3
YOLO-v8n (this work)	64.5	48.6	58.3	75.5	69.7	$56.2^{+7.6\%}$	57.7	76.8	0.3	0.3	6.0
YOLO-v5s (this work)	64.3	49.3	58.1	76.4	69.2	$56.6^{+7.3\%}$	58.1	76.4	0.4	0.5	17.7
YOLO-v5s6u (this work)	66.1	47.4	61.4	73.4	**71.0**	53.8	**65.2**	70.7	0.4	0.4	29.6
YOLO-v8s (this work)	64.9	50.3	59.3	76.1	70.6	$\mathbf{58.8}^{+8.5\%}$	59.2	76.7	0.55	0.5	21.5

The tuning of the confidence parameter is very surprising: a general increase of 10% in terms of mAP50 and mAP50-95 is observed.

In this context, the performance of the other YOLO v5x models are achieved by a YOLO v5 or v8 nano - YOLO v5x is 20x larger than a nano model.

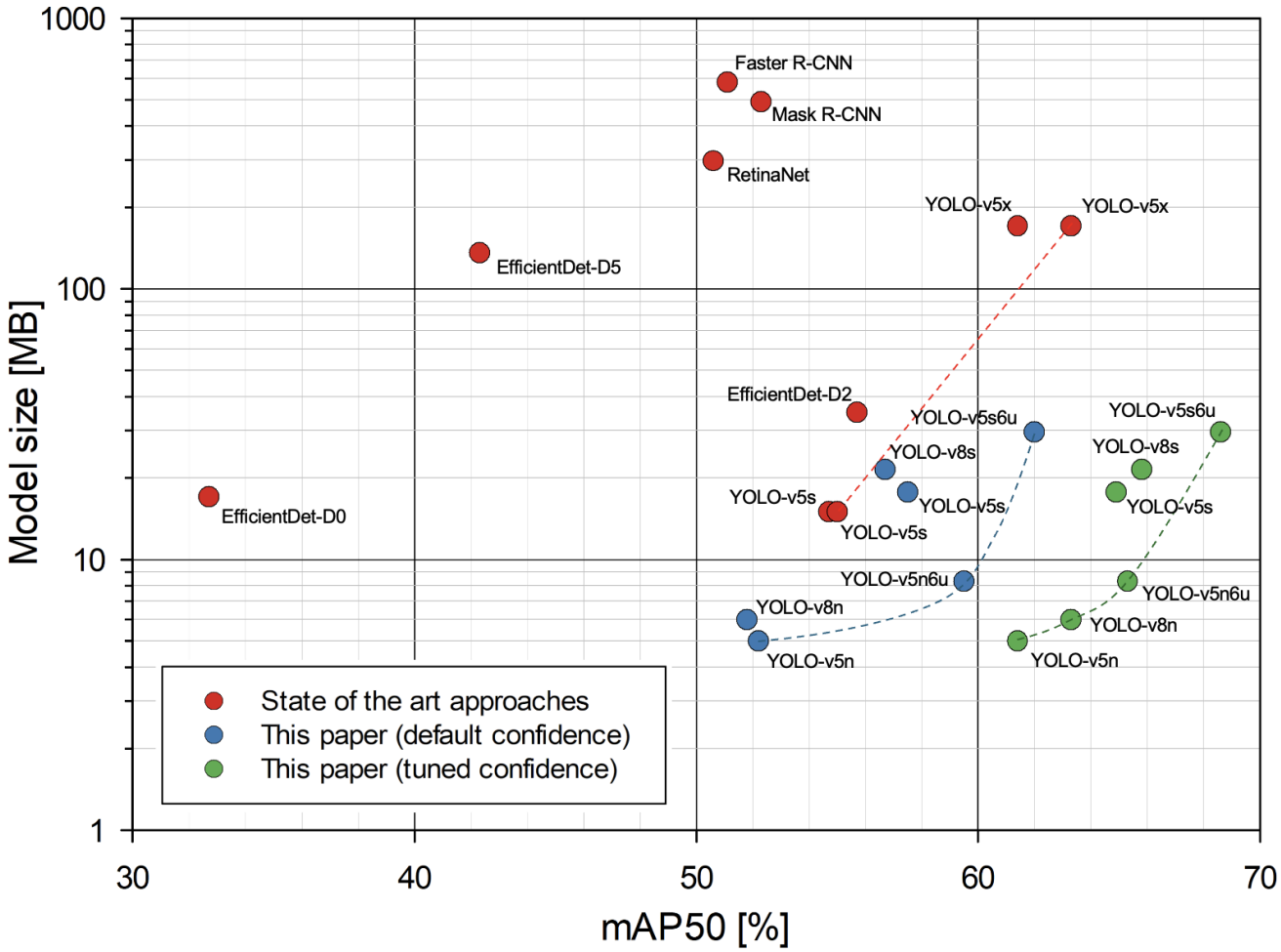
Our YOLO v5s outperforms the others YOLO v5s [Cordova2022litter] [Das2023outdoor] by 10% in terms of mAP50 and mAP50-95. Unfortunately, the data provided by the other papers are incomplete and not comparable in terms of recall and precision, so only an internal comparison was made.

YOLO v8 nano and small models benefit the most from tuning the Confidence and IoU thresholds, achieving an approximately 10% increase in performance without a decrease in recall.

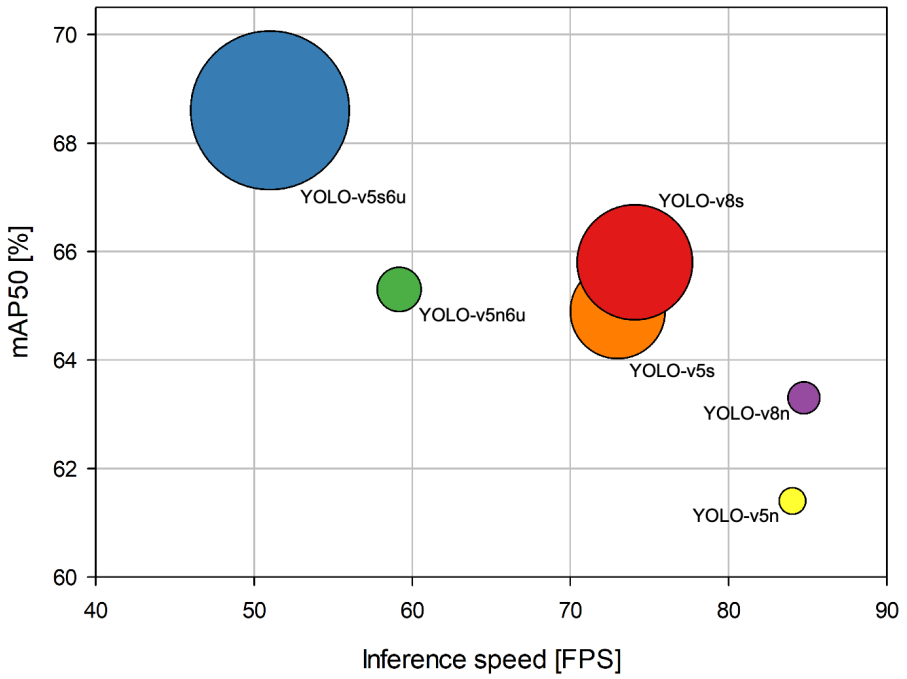
YOLO v5s6u can be considered the best model overall, as it excels in object detection due to its higher recall and mAP50, although the increase recorded with tuning is not very substantial.

YOLO v5n6u represents the best balanced model: it is only slightly heavier than the nano models yet demonstrates performance similar to the larger YOLO v5s6u, particularly in terms of recall and mAP50, with an even greater mAP50-95. The small versions of YOLO v5 and v8 are more precise, but the benefits they provide are not significant enough to meet the primary objectives.

To better understand the differences in model size, the following figure illustrates the substantial performance gap between the models and the state of the art, highlighting how better performance can be achieved with significantly lighter models.



For an internal comparison of the models used, the following figure clearly illustrates their ability to run on devices with low computational capabilities.



YOLO v5n6u and v5s6u have longer inference speeds, primarily due to the input image size being twice the standard size. At the same time, the size difference between the Nano and Small model versions is visible, impacting the storage space required on the final device.

However, in both cases, the models are suitable for embedded devices like smartphones. The YOLO v5s6u, could be considered borderline for this application, especially since its performance does not significantly surpass that of its competitors to justify the increased resource requirements.

Since our final goal is to run the trained models on edge devices, as a final step we converted our PyTorch models into TensorFlow Lite format with different quantizations and measuring the final model size as well as its performance on the TACO-1 task.

The experimental results are reported in the following table.

Model	Format	Default conf. mAP50	Default conf. mAP	Tuned conf. mAP50	Tuned conf. mAP	Size (MB)
YOLO-v5n	.pt	52.2	34.3	61.4	47.2	5.0
	float32.tflite	51.7	33.7	60.9	46.8	9.8
	float16.tflite	51.8	33.7	61.0	46.8	5.0
	integer quant.tflite	44.3	27.1	56.7	41.8	2.5
	full integer quant.tflite	43.8	27.0	57.0	41.7	2.5
	int8.tflite	51.0	33.2	60.2	46.7	2.7
YOLO-v5n6u	.pt	59.5	41.7	65.3	50.0	8.3
	float32.tflite	59.4	41.3	65.1	49.3	16.7
	float16.tflite	59.4	41.4	65.1	49.4	8.4
	integer quant.tflite	50.0	30.3	56.3	37.0	4.3
	full integer quant.tflite	49.2	29.8	55.6	36.6	4.3
	int8.tflite	59.2	40.8	65.3	49.3	4.8
YOLO-v8n	.pt	51.8	34.7	63.3	48.4	6.0
	float32.tflite	51.7	34.7	61.8	47.4	11.8
	float16.tflite	51.7	34.7	61.8	47.4	5.9
	integer quant.tflite	44.9	27.8	58.6	41.5	3.0
	full integer quant.tflite	45.3	28.4	59.3	42.5	3.0
	int8.tflite	51.8	34.1	63.0	47.8	3.1
YOLO v5s	.pt	57.7	39.1	64.9	52.2	17.7
	float32.tflite	57.7	38.8	64.9	51.5	35.1
	float16.tflite	57.7	38.7	64.9	51.5	17.6
	integer quant.tflite	49.2	29.4	60.4	42.3	8.9
	full integer quant.tflite	49.8	29.9	61.9	43.4	8.9
	int8.tflite	56.6	37.4	64.1	50.1	9.0
YOLO-v5s6u	.pt	62.0	40.7	68.6	49.8	29.6
	float32.tflite	62.2	40.6	68.3	49.5	59.2
	float16.tflite	62.1	40.6	68.3	49.6	29.7
	integer quant.tflite	46.3	26.2	54.6	35.6	14.9
	full integer quant.tflite	45.2	25.9	53.3	35.4	14.9
	int8.tflite	58.6	37.7	62.1	45.2	15.5
YOLO-v8s	.pt	56.7	39.7	65.8	54.3	21.5
	float32.tflite	56.7	39.9	66.0	54.2	42.8
	float16.tflite	56.5	39.9	66.0	54.2	21.4
	integer quant.tflite	48.3	30.0	61.1	44.9	10.8
	full integer quant.tflite	48.6	29.4	61.4	43.8	10.8
	int8.tflite	54.6	36.8	62.5	50.3	10.9

The different post-training quantization schemes here considered range from a simple conversion of the model into TFLite file as full-precision floating point (float32.tflite) to half-precision floating point (float16.tflite), from the quantization as 8-bit integers of only the weights of the model (integer_quant.tflite) also exploiting Dynamic Range Quantization (int8.tflite), to the quantization as 8-bit integers of both the weights and the activations (full_integer_quant.tflite).

From the results reported in the previous table we can observe how if the inference has to be done on an edge TPU (e.g., Google Coral) where a full integer quantization of both weights is required, we can obtain a model that on average is about 50% of the original model with an average reduction in mAP50 of about 6.8%. Instead, if the edge device has a GPU that can be used for inference (e.g., ARM Mali, Qualcomm Adreno, etc.), a FP16 quantized model permits to limit the average degradation in mAP50 to just 0.4% at the cost of a model having the same size as the original one.

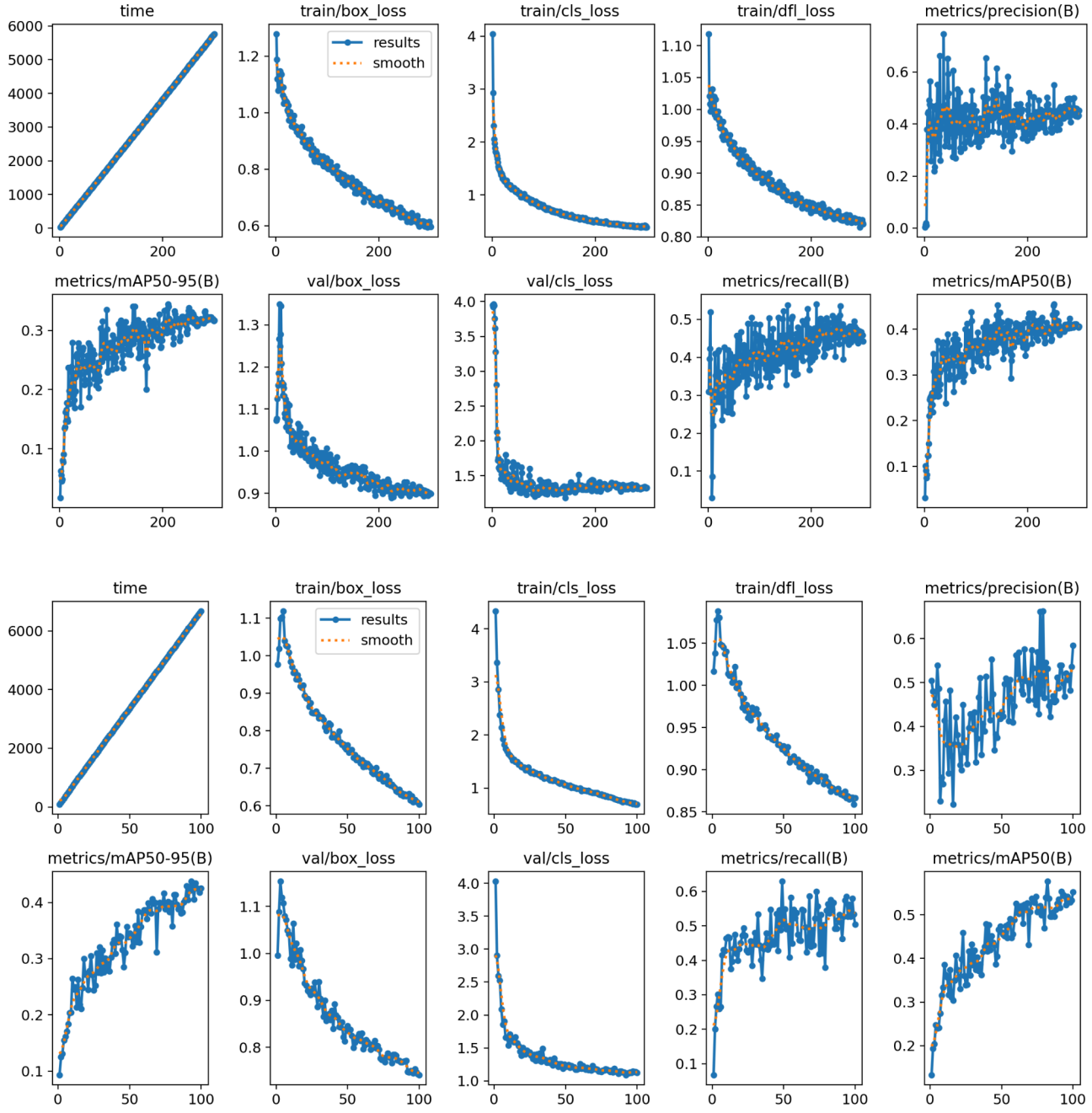
Volume estimation on the UniMiB Trash dataset

This project also aims to explore whether the task can be approached in a multiclass context, providing additional information beyond the number of waste items detected. For example, the priority may differ significantly if there are 10 large rubbish bags in an image compared to 10 cigarette butts. To address this, an effort is made to categorize waste based

on size: large, medium, small, and micro, with the last included for dataset completeness, despite being of lesser relevance.

The UniMiB Trash dataset is initially partitioned with an 80-20% split between training and testing, and the training set is further divided into 70% for training and 10% for validation. Class balance is maintained across all partitions to ensure a fair distribution of samples.

The model training setup follows the standard YOLO configuration, with the only difference being an increased total number of epochs to 300. It was observed that, compared to using fewer epochs, the models become more recall oriented and more resistant to increases in the confidence parameter for threshold tuning, probably due to the ability to predict with higher confidence scores. Some training results are shown in the following figure.



Results on the test are shown in the following table.

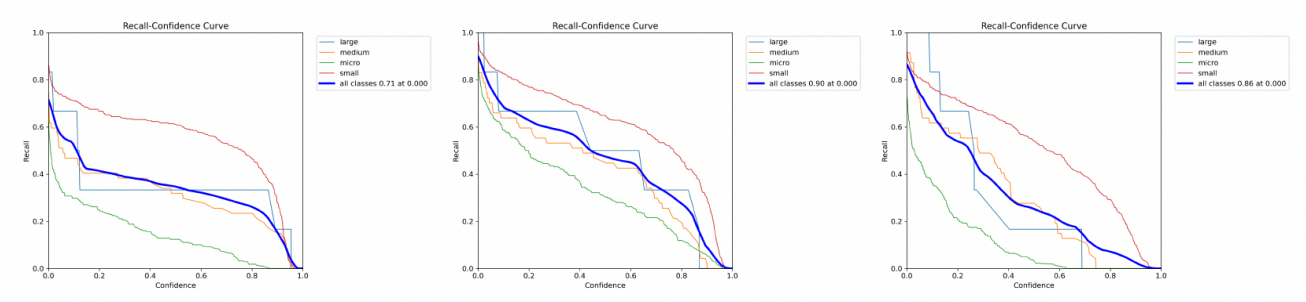
Method	Class	Default thresholds mAP50	mAP50-95	recall	precision	Tuned thresholds mAP50	mAP50-95	recall	precision	Tuned Value Conf	Iou	Size (MB)
YOLO-v5n (this work)	average	42.6	31.2	43.9	51.8	44.5	33.8	43.1	53.2	0.1	0.4	**5.0**
	large	44.1	30.2	54.5	47.8	44.2	30.2	50.0	51.1			
	medium	36.0	28.5	42.9	47.9	39.3	32.1	43.5	49.8			
	small	69.4	54.6	66.7	68.8	71.9	58.7	66.3	70.2			
	micro	20.8	11.7	11.3	42.8	22.7	14.2	12.5	41.8			
YOLO-v5n6u (this work)	average	53.3	42.0	49.9	58.7	55.9	**45.7**	**58.9**	53.7	0.2	0.5	8.3
	large	48.1	36.7	50.0	48.9	54.5	42.8	63.6	49.2			
	medium	37.9	31.7	43.8	43.5	39.6	34.8	49.3	43.0			
	small	77.4	65.0	70.1	**76.4**	**78.4**	67.9	**75.9**	69.0			
	micro	49.9	34.5	35.8	**65.9**	51.3	37.4	46.7	53.5			
YOLO-v8n (this work)	average	44.4	33.0	47.2	51.0	46.7	36.9	46.7	51.9	0.2	0.3	6.0
	large	51.3	39.0	59.1	55.2	48.7	38.4	54.5	57.1			
	medium	34.0	25.7	35.6	48.4	41.8	32.7	35.6	48.1			
	small	68.7	54.7	69.9	64.4	70.6	59.3	70.1	66.3			
	micro	23.4	12.7	24.2	35.8	28.1	17.2	26.7	35.8			
YOLO-v5s (this work)	average	44.0	32.1	40.6	58.9	45.6	35.3	47.8	47.6	0.1	0.2	17.7
	large	37.5	24.8	40.9	52.4	40.3	26.5	40.9	54.6			
	medium	38.7	33.4	42.5	35.4	39.0	33.6	42.8	39.0			
	small	73.0	61.5	73.1	65.4	73.1	61.6	73.0	66.1			
	micro	33.1	21.6	34.8	37.0	33.2	21.6	35.7	36.5			
YOLO-v5s6u (this work)	average	54.4	41.0	54.8	58.8	**56.1**	44.9	55.2	**60.7**	0.35	0.5	29.6
	large	43.2	28.7	54.5	49.7	44.7	31.7	54.5	55.7			
	medium	42.4	31.6	45.2	49.9	**45.8**	**37.0**	45.2	**51.3**			
	small	78.0	65.9	71.6	74.9	77.1	**68.6**	72.2	75.2			
	micro	54.0	37.8	47.8	60.6	**56.6**	**42.1**	**48.8**	60.5			
YOLO-v8s (this work)	average	44.6	32.4	48.9	45.3	47.4	37.6	46.5	49.3	0.2	0.6	21.5
	large	52.4	37.1	50.0	59.3	**56.8**	**45.4**	45.5	**66.7**			
	medium	28.1	21.6	38.4	29.9	29.5	25.1	38.4	32.6			
	small	69.1	54.6	73.3	56.2	70.2	58.4	71.9	60.2			
	micro	28.8	16.3	33.9	35.7	32.9	21.5	30.1	37.7			
YOLO-World X (zero shot)	average	14.8	9.4	48.8	15.0	15.8	10.2	55.7	12.3	0.01	def	140.0
	large	8.3	6.4	**81.8**	2.2	7.5	5.7	72.7	1.5			
	medium	15.7	12.3	76.7	6.6	15.8	12.3	**80.8**	5.4			
	small	33.8	18.4	34.5	49.6	37.3	21.7	56.9	38.0			
	micro	1.5	0.4	2.3	1.7	2.6	0.9	12.2	4.6			

Considering that the dataset is not very large, with just over 1,200 images, and exhibits significant class imbalance, the performance is reasonable and roughly comparable to that of the 1-class TACO dataset.

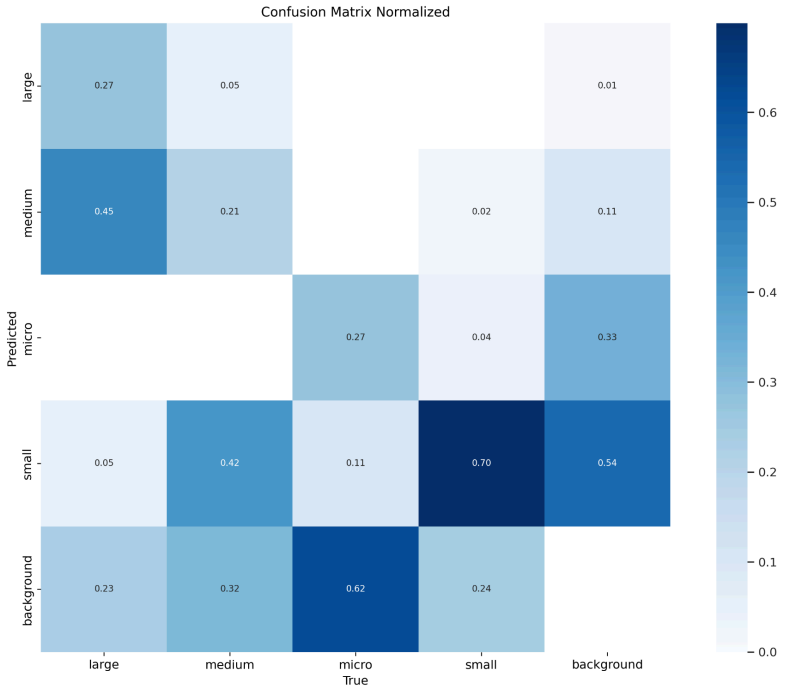
Additionally, the *micro* class occasionally contributes to lower overall performance, although it is not a primary focus for this task. Anyway, the results for the micro class are much better with YOLO v5n6u and v5s6u compared to the other models, due to the double resolution of the input images.

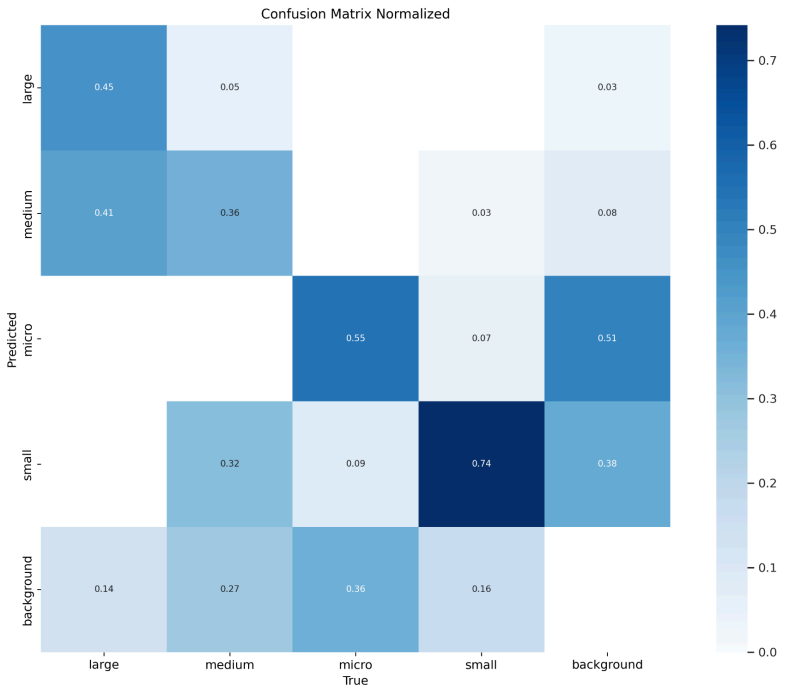
The *small* class generally achieves robust performance on all models, the medium class is the one on which we need to look for more work.

With a higher number of classes, tuning the Confidence and IoU thresholds becomes more challenging. As shown in the following figure, detecting the micro class is particularly difficult for YOLO models when dealing with such small objects. However, this is not the case for YOLO v5s6u (center plot), which performs significantly better and demonstrates good performance, likely due to the higher resolution of the input images.



The confusion matrices shown in the following figure facilitate the analysis of predictions on a class-by-class basis, revealing potential issues.





The challenges with the micro class arise from non-detections instead of a poor separation between classes, because many micro instances are not detected rather than classified into other categories.

Specifically, for YOLO v5s6u (in the upper part of the previous figure), 55% of instances are accurately identified and classified; only 9% are misclassified as small, while 36% remain unidentified. Additionally, there are numerous erroneous predictions of background elements being classified as *micro* instances. This distinction is further supported by the observation that only 7% of *small* instances are misclassified as *micro*. Conversely, the detection rates for *large* and *medium* classes are more favorable; for example, only 14% of *large* instances are not detected. However, 41% of the detected *large* instances are misclassified as *medium*. In the *medium* class, instances are typically not confused with the *large* class, but 36% are misclassified as *small*. Very few background instances are misclassified as *large* (3%) or *medium* (8%). The *small* class, which is the most prevalent, exhibits strong performance, demonstrating effective separation from the other classes and relatively low misclassification rates (16%).

The previous results are further confirmed by analysing the performance of YOLO v8s (in the bottom part of the previous figure). However, it is important to note that this model exhibits significantly lower detection capacity for instances across each class compared to YOLO v5s6u. Specifically, there is a detection gap of 5% in the *medium* class and a striking 26% in the *micro* class, highlighting a notable decline in performance.

Although the *micro* class's results are less favorable, it's worth mentioning that this class holds relatively minor importance for the overarching objectives of the study. Furthermore, the number of True Positives has also decreased in the *large* class, with a reduction of 18%, and in the *medium* class, which has decreased by 15%. Conversely, the *small* class shows a much smaller decline of only 4%.

In terms of classification capabilities, the models exhibit roughly similar performance, though with a slight overall decrease in effectiveness. Additionally, there has been an increase in the misclassification rates, with *large* instances being incorrectly classified as *medium* by 4%, and *medium* instances misclassified as *small* by 10%. This misclassification trend is concerning, as it can lead to misunderstandings of the data. Moreover, the instances predicted as *small* that are confused with the background have increased by 16%. In contrast, the *micro* class has seen a decrease in misclassification, which can be attributed to the model's reduced ability to predict instances of this class accurately.

Therefore, it is essential to closely examine the differences between the two models using the Confusion Matrix. The utilization of YOLO v5s6u, which processes images at a resolution of 1280 pixels, does not merely enhance the performance of the litter class. It also contributes to a more robust overall detection capacity for the other classes. This improvement is reflected in the superior classification abilities across all four classes in the dataset, particularly for larger waste items. The findings suggest that using higher-resolution input images can significantly benefit the model's performance, facilitating more accurate object detection and classification.

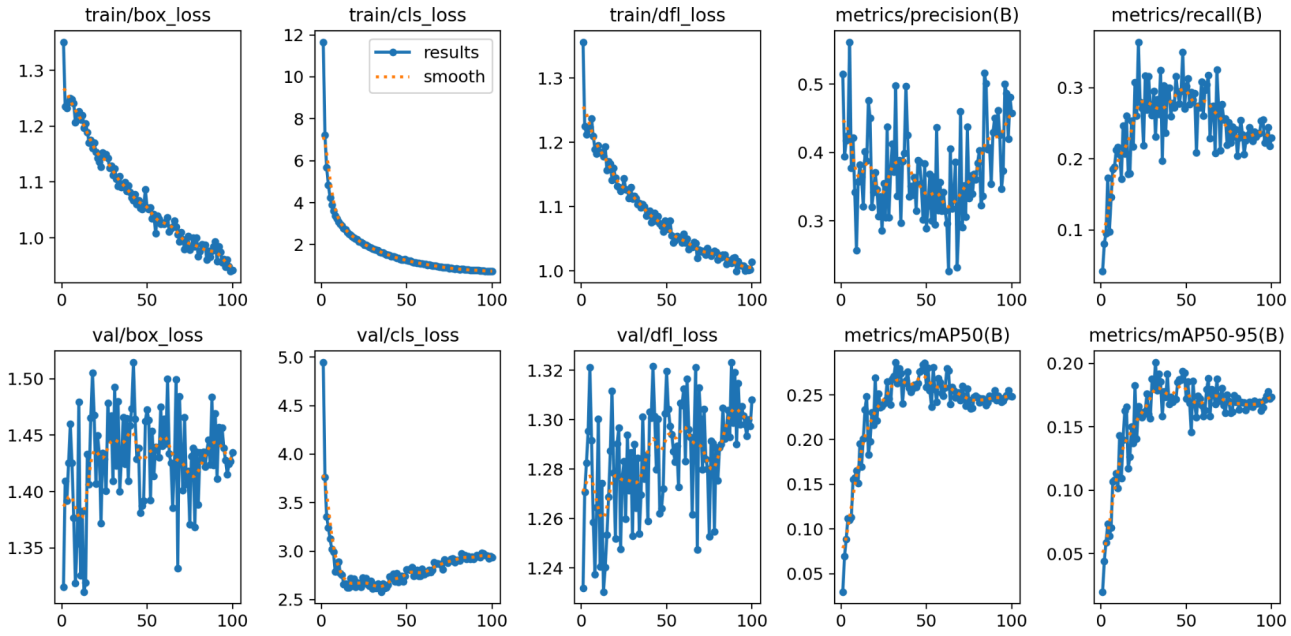
Material segmentation on TACO dataset

The TACO dataset is distributed with the annotation of 60 different waste categories. Due to the difficulty posed by the extremely unbalanced classes, researchers have usually used TACO-1 (i.e., all 60 classes are merged to a single class representing litter), or TACO-10 where litters are grouped in ten macro categories depending on the litter type. This version

has the disadvantage that for example glass bottles and plastic bottles are merged together in the macro class bottle, therefore losing the material information.

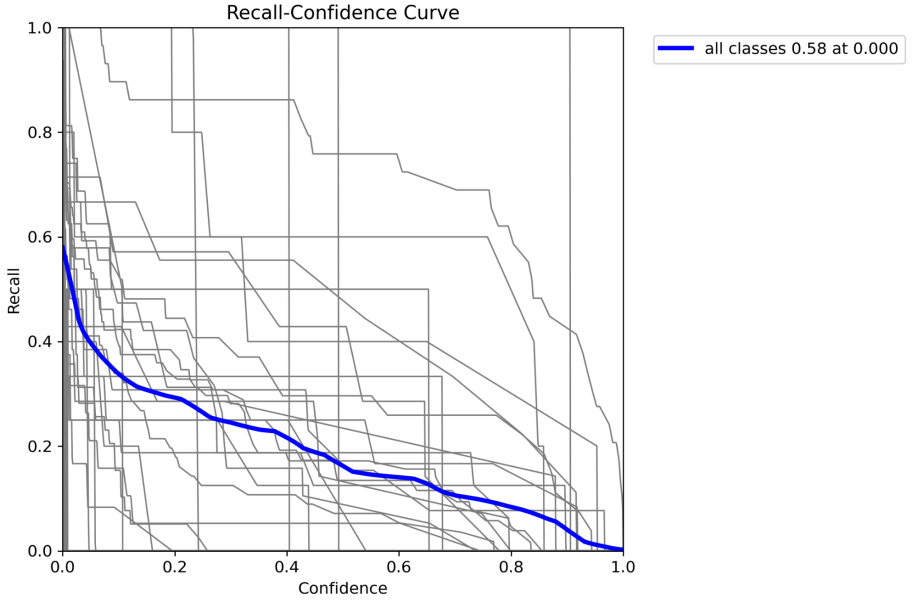
Since we are interested in the litter material instead, we experiment with the original TACO.

The training plots of YOLO v5s6u are reported in the figure below.



The trained detector is able to reach a mAP50 of 28.6, while the only other result reported in the literature is 25.5 from [Das2023outdoor].

In the following figure we also report the recall-confidence plot, from which it is possible to see that some categories are not detected, mainly due to the fact that very few examples are available.



5 Conclusions and remarks

This deliverable presents the outcomes of developing and evaluating lightweight models for waste recognition, volume estimation, and material segmentation in the context of the COBOL project. The key achievements and insights can be summarized as follows:

1. **Lightweight Model Development**:
    - By employing YOLO-v5 and YOLO-v8 models, focusing on their tiny and small variants, we achieved competitive performance on the TACO dataset. These models balance detection accuracy and resource efficiency, making them suitable for deployment on edge devices such as smartphones and embedded systems.
    - Fine-tuning hyperparameters, particularly confidence and IoU thresholds, resulted in substantial improvements in performance, highlighting the importance of post-training optimization for lightweight models.
2. **Dataset Contributions**:
    - The UniMiB Trash dataset was collected and annotated, emphasizing size-based categorization to prioritize waste of varying significance. With over 1,200 high-resolution images and more than 6,000 annotated instances, the dataset provides a unique resource for evaluating waste detection models under realistic conditions.
    - The dataset's focus on abandoned waste in natural settings and its inclusion of complex scenarios ensures its applicability to real-world challenges.
3. **Evaluation and Results**:
    - The YOLO-v5s6u model emerged as the best-performing lightweight detector, achieving high mAP scores and recall rates while maintaining a balance between precision and inference speed.
    - Quantization techniques, including FP16 and INT8, further reduced model sizes, making deployment on edge TPU and GPU devices feasible without significant degradation in detection performance.
4. **Insights from Volume Estimation and Material Segmentation**:
    - Initial efforts in volume estimation demonstrated the potential to prioritize waste types based on size, enhancing the utility of waste detection systems in urban and rural environments.
    - Material segmentation on the TACO dataset achieved promising results, surpassing previous benchmarks, despite the challenges posed by imbalanced class distributions.

6 References

[Bianco2024efficient] Bianco, S., Gaviraghi, E., & Schettini, R. (2024, September). Efficient Deep Learning Models for Litter Detection in the Wild. In 2024 IEEE 8th Forum on Research and Technologies for Society and Industry Innovation (RTSI) (pp. 601-606). IEEE.

[Cordova2022litter] Córdova, M., Pinto, A., Hellevik, C. C., Alaliyat, S. A. A., Hameed, I. A., Pedrini, H., & Torres, R. D. S. (2022). Litter detection with deep learning: A comparative study. Sensors, 22(2), 548.

[Das2023outdoor] Das, D., Deb, K., Sayeed, T., Dhar, P. K., & Shimamura, T. (2023). Outdoor Trash Detection in Natural Environment Using a Deep Learning Model. IEEE Access.

[Redmon2016look] Redmon, J. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition.

[yolov5_ultralytics] Jocher, G. (2020). YOLOv5 by Ultralytics. 2020. URL https://github. com/ultralytics/yolov5.

[yolov8_ultralytics] Jocher, G., Chaurasia, A., Qiu, J., (2023). YOLOv8 by Ultralytics. 2023. URL https://github. com/ultralytics/ultralytics.

[Zou2023object] Zou, Z., Chen, K., Shi, Z., Guo, Y., & Ye, J. (2023). Object detection in 20 years: A survey. Proceedings of the IEEE, 111(3), 257-276.