Efficient Deep Learning Models for Litter Detection in the Wild

Simone Bianco, Elia Gaviraghi, Raimondo Schettini
Department of Informatics, Systems and Communication
University of Milano-Bicocca
Milano, Italy

***Abstract*—Littering presents a substantial environmental hazard and impacts our well-being. The importance of automatic litter detection lies in its ability to identify waste in the environment, thereby enhancing the efficiency of subsequent waste management operations. In order to achieve a comprehensive and detailed survey of an area for litter detection, one of the most effective approaches is to utilize the collective efforts of citizen science. In this work we assess the performance of the most efficient object detection methods aiming their use in the type of devices typically employed in citizen science activities, e.g. smartphones with low processing capabilities. Experiments on the Trash Annotations in COntext (TACO) dataset show that by exploiting our training procedure, the efficient models that we tested are able to surpass the performance reached by larger models in the state of the art. Moreover, experiments show the among the efficient object detectors tested, the *small* model variants offer the best trade off between model size and litter detection performance.**

***Index Terms*—Litter detection, efficient detector, YOLO, citizen science**

I. 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 [1].

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.

Among the different existing datasets (e.g., [2]–[5]), the TACO (Trash Annotations in Context) dataset [6] 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 have 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 paper, 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 [7] and YOLO-v8 [8] 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 [9] to investigate the trade-off between model size and detection performance.

II. Related Works

Existing solutions in the state of the art differ both in terms of architecture of the adopted litter detector model, in terms of the dataset(s) used, and in terms of how the problem is casted, i.e., detection or segmentation.

Proença and Simoes presented TACO (Trash Annotations in Context) dataset [6], and adopted a Mask-RCNN to be used as a baseline. Wang et al. [2] described the creation of the MJU-WASTE dataset and tested several models with high number of parameters and various backbones are used for segmentation task (FCN-8s, PSPNet, CCNet and DeepLabv3). Patrizi et al. [10] introduced a data-augmentation procedure to expand existing datasets by cropping solid waste in images taken on a uniform white background and superimposing it on more realistic backgrounds. Córdova et al. [3] compared several state-of-the-art CNN architectures (e.g., Faster RCNN, Mask-RCNN, EfficientDet, RetinaNet, YOLO-v5s and YOLO-v5x) on two litter image datasets. Majchrowska et al. [11] merged collections from open-source datasets and proposed a two-stage detector for litter localization (based on EfficientDet-D2) and classification (based on EfficientNet-B2). Jalal et al. [12] compared different YOLO-v5 variants (from YOLO-v5s to YOLO-v5x) on a custom dataset. Das et al. [13] also compared different YOLO-v5 variants (from YOLO-v5s to YOLO-v5x), including a test time augmentation (TTA) step to increase model inference accuracy at the cost of longer inference time. Mandhati et al. [14] used images from TACO

dataset together with the PlastOPol [3], UAV-DB [4] and UAVVaste [5] datasets, on which they evaluated YOLO-v5l and Faster R-CNN.

III. Methods

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 [15]. YOLO belongs to the category of one-stage detectors [16] and spatially separates bounding boxes and associates probabilities to each of the detected object 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 [**?**] and YOLO-v8 [8], focusing in particular on the *tiny* and *small* models.

Both YOLO-v5 and YOLO-v8 use input images of size 640×640, while a variant of YOLO-v5, available in both tiny (YOLO-v5n6u) and small (YOLO-v5s6u) model sizes, uses input images of size 1280×1280.

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, and *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 of 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 Figure 1 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.

Fig. 1. Qualitative evaluation of the effect of the modification of the confidence threshold from the default value $conf = 0.001$ (left) to a higher value $conf = 0.5$ (right).

IV. Experimental results

A. Dataset and metrics

In this paper we selected TACO dataset [6], that contains 1500 images with a total of 4784 annotations. TACO objects are labeled into 60 categories that can be grouped in 28 super categories, including the category *Unlabeled waste* for hard to recognize or heavily occluded objects; some of these super categories are over-represented (e.g., *cigarettes* and *unlabeled waste*) while others are under-represented - there are 17 super categories with less than 10 entries each. In the TACO paper a 10-class subdivision is proposed, keeping 9 of the original super categories and merging the remaining ones into a single class called *Other*; neither this subdivision, however, does result in a good balance of the dataset. To mitigate this problem, in the TACO paper also a 1-class subdivision is proposed, named TACO-1, which is the most frequently used annotation where only one class is considered, i.e. the *litter* class. Independently from the number of classes considered, in general TACO is a very challenging dataset due to the presence of very small objects (e.g., cigarette butts, bottle caps, rope, strings, etc.) and transparent objects (e.g., bottles, glass, etc.). In Figure 2 we report the histogram and the cumulative distribution of the length of the diagonal of the bounding boxes relative to the longest image side. From the plots it can be noticed that the most frequent bins are the smallest ones; in particular, the most frequent bin corresponds to a bounding box having the longest side of about 12 pixels inside an image scaled with the longest side equal to 640, confirming the difficulty of the dataset.

The different methods are compared in terms of:

- mAP50, which measures the detection effectiveness in terms of mean average precision (AP) at the intersection over union (IoU) of 50%;
- mAP50-95, which measures the detection effectiveness in terms of mean average precision considering the average of 10 precision values computed changing the IoU value from 50% to 95%, at steps of 5%.
- the efficiency in terms of model size in Megabytes (MB)

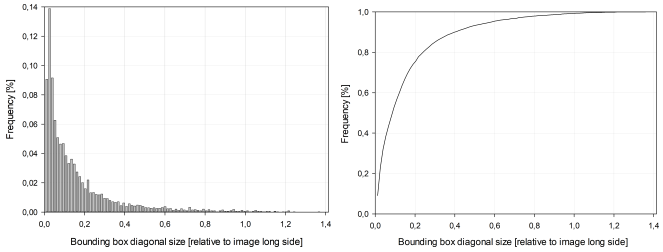


Fig. 2. Histogram (left) and cumulative distribution (right) of the annotated bounding boxes on the TACO dataset measured as the size of the diagonal of the bounding box relative to the image long side.

and processing time in frames per second (FPS).

B. Experimental setup

Unfortunately TACO dataset does not came with a standard partitioning into training, validation and test sets. The common procedure is to randomly split the dataset into 80% for training and 20% for testing. Then, some researchers [3] apply a 5-fold cross validation procedure and select the model to be tested as the one with the highest performance on the respective validation fold. In order to have comparable results with the state of the art, in this paper we split the dataset into 70% for training, 10% for validation, and 20% for testing. Then we select the model to be tested as the one with the highest performance in terms of mAP50 on the validation set.

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.

The training is performed using Python-3.10.12 and Ultralytics YOLOv8.2.7, and PyTorch-2.1.0 libraries using a single NVIDIA GeForce GTX 1080 GPU with 8GB RAM.

C. Experimental results

The results of the tested models on the TACO-1 task are reported in Table I and compared with the state of the art; the performance are reported in terms of mAP50, mAP50-95 and model size. Performance of the methods in the state of the art are taken from the respective papers and unfortunately, not all of them report the mAP50-95 performance. Concerning our models, their performance is reported using the default confidence and our tuned confidence. The same information is also plotted in Figure 3.

From the reported results it is possible to notice how all the methods benefited from the confidence tuning, with an average improvement of 8.2% in mAP50 and 12.0% in mAP50-95, with YOLO-v8 benefiting the most from this tuning.

Unsurprisingly, the best results among the nano models is obtained by YOLO-v5n6u and among the small ones by YOLO-v5s6u, that work on input images with size 1280 and are able to better detect small objects. This is also confirmed in Figure 5 where the detections of our models on a set of images belonging to the test set are reported.

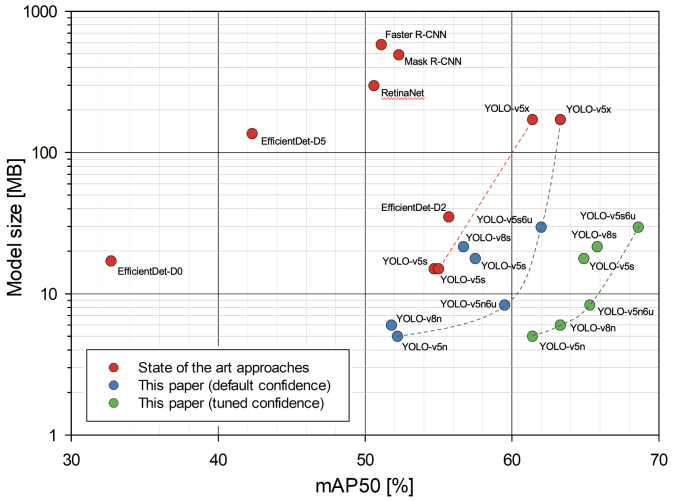


Fig. 3. Performance comparison on the TACO dataset on the TACO-I task in terms of mAP50 and model size for approaches in the state of the art (red circles), our models with default confidence (blue circles), and our models with tuned confidence (green circle). For each group of models, the corresponding performance Pareto front is reported with a dashed line having the same color.

In comparison with the state of the art solutions, we can see in Figure 3 that we can push the Pareto front of the performance towards the bottom right corner, i.e. towards the region of smaller models able to reach higher mAP50 values. For example our best model, YOLO-v5s6u outperforms the best method in the state of the art (YOLO-v5x) by 5.3% in mAP50 being just 17.3% of its size. At the same time, YOLO-v8n reaches the same mAP50 of the best method in the state of the art being just 3.5% of its size. In Figure 4 we also report a comparison in terms of mAP50, inference speed (in FPS), and model size for our models tested in our hardware configuration. Form the plot it is possible to see how all the models are able to reach super real-time performance, with the *small* models (YOLO-v5s and YOLO-v8s) offering the best trade-off between inference speed and detection accuracy.

Since our final goal is to run the trained models on edge devices, as final experiment 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 Table II.

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 Table II 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%

TABLE I
LITTER DETECTION RESULTS ON TACO DATASET ON THE TACO-1 TASK. THE BEST RESULTS ARE REPORTED IN BOLD.

Method	Train / test dataset	Default confidence mAP50	Default confidence mAP50-95	Tuned confidence mAP50	Tuned confidence mAP50-95	Improvement mAP50	Improvement mAP50-95	Size (MB)
RetinaNet [3]	TACO	50.6						297.0
Faster R-CNN [3]	TACO	51.1						580.0
Mask R-CNN [3]	TACO	52.3						491.0
EfficientDet-D0 [3]	TACO	32.7						17.0
EfficientDet-D2 [11]	Extended TACO	56.8	40.4					35.0
EfficientDet-D5 [3]	TACO	42.3						136.0
YOLO-v5s [3]	TACO	54.7						15.0
YOLO-v5s [13]	TACO	55.0	38.5					15.0†
YOLO-v5x [3]	TACO	**63.3**						171.0
YOLO-v5x [13]	TACO	61.4	**47.6**					171.0†
YOLO-v5n (this paper)	TACO	52.2	34.3	61.4	47.2	+9.2	+12.9	**5.0**
YOLO-v5n6u (this paper)	TACO	59.5	41.7	65.3	50.0	+5.8	+8.3	8.3
YOLO-v8n (this paper)	TACO	51.8	34.7	63.3	48.4	**+11.5**	+13.7	6.0
YOLO-v5s (this paper)	TACO	57.7	39.1	64.9	52.2	+7.2	+13.1	17.7
YOLO-v5s6u (this paper)	TACO	62.0	40.7	**68.6**	49.8	+6.6	+9.1	29.6
YOLO-v8s (this paper)	TACO	56.7	39.7	65.8	**54.3**	+9.1	**+14.6**	21.5

† not declared by the authors, taken from [3]

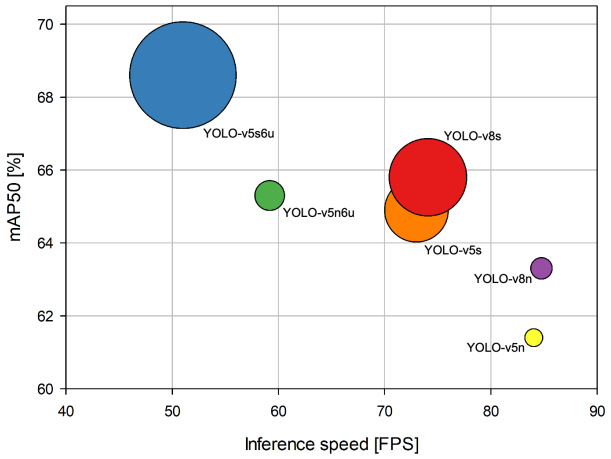


Fig. 4. Performance comparison on the TACO dataset on the TACO-I task in terms of mAP50, inference speed and model size for our models with tuned confidence.

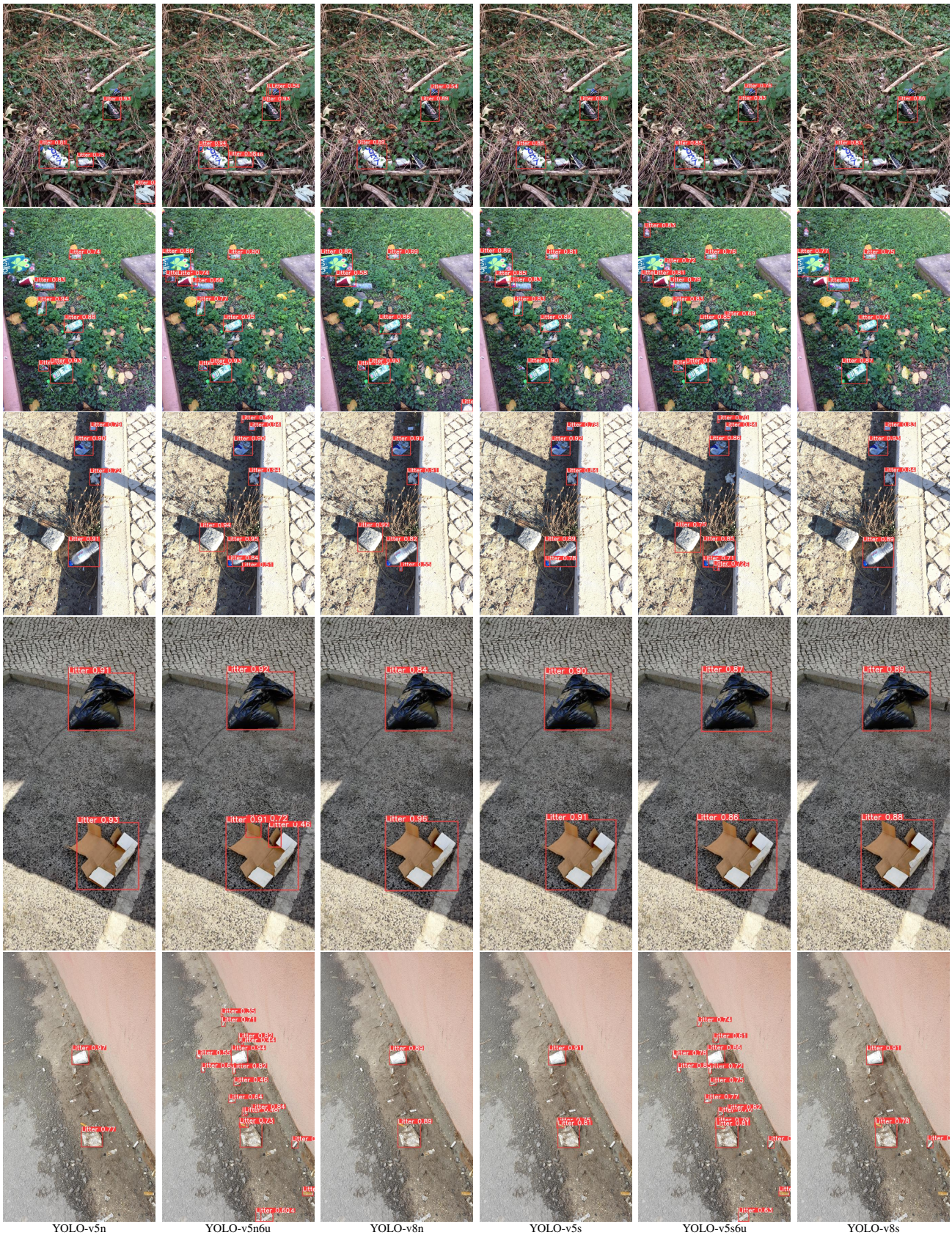
of the original model with an average reduction in mAP50 of about 6.8%. Instead, if the edge device has a GPU that can be used for inference (e.g., ARM Mali, Qualcomm Adreno, etc.), a FP16 quantized model permits to limit the average degradation in mAP50 to just 0.4% at the cost of a model having the same size as the original one.

V. CONCLUSION

In this work, we assessed the performance of the most efficient object detection methods, specifically targeting their use on devices typically employed in citizen science activities, such as smartphones with low processing capabilities. Our experiments on the Trash Annotations in Context (TACO) dataset demonstrate that by exploiting our training procedure, the efficient models we tested surpass the performance of the best model in the state of the art, with an improvement up to 5.3% in terms of mAP50 with a model that is 17.3% of its size.

TABLE II
LITTER DETECTION RESULTS ON TACO DATASET ON THE TACO-1 TASK FOR THE MODELS CONVERTED INTO TENSORFLOW LITE FORMAT.

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



YOLO-v5n YOLO-v5n6u YOLO-v8n YOLO-v5s YOLO-v5s6u YOLO-v8s

Fig. 5. Litter detection results of our models on some sample images belonging to the test set of the TACO dataset. Detected litter is annotated with a red bounding box reporting also its detection confidence.

The experimental result also indicate that among the efficient object detectors tested, the *small* model variants offer the best trade-off between model size and litter detection performance. We also showed that by exporting the trained models with different quantization levels it is possible to further reduce the model size at at about 52% of its original size at the cost of an average reduction in detection performance of about 2% in both mAP50 and mAP50-95.

These results underscore the potential of deploying lightweight yet effective object detection models on resource-constrained devices used by citizen scientists. This approach not only supports better waste management practices by enabling more accurate and widespread litter detection but also empowers communities to actively participate in environmental conservation efforts.

ACKNOWLEDGMENT

This work has been supported the COmmunity-Based Organized Littering (COBOL) national research project, which has been funded by the MUR under the PRIN 2022 PNRR program (contract nr. P20224K9EK).

REFERENCES

[1] L. Du, H. Xu, and J. Zuo, "Status quo of illegal dumping research: Way forward," *Journal of Environmental Management*, vol. 290, p. 112601, 2021.

[2] T. Wang, Y. Cai, L. Liang, and D. Ye, "A multi-level approach to waste object segmentation," *Sensors*, vol. 20, no. 14, p. 3816, Jul. 2020. [Online]. Available: http://dx.doi.org/10.3390/s20143816

[3] M. Córdova, A. Pinto, C. C. Hellevik, S. A.-A. Alaliyat, I. A. Hameed, H. Pedrini, and R. d. S. Torres, "Litter detection with deep learning: A comparative study," *Sensors*, vol. 22, no. 2, p. 548, 2022.

[4] R. Hann and J. Wallisch, "UAV Database," 2020. [Online]. Available: https://doi.org/10.18710/L41IGQ

[5] M. Kraft, M. Piechocki, B. Ptak, and K. Walas, "Autonomous, onboard vision-based trash and litter detection in low altitude aerial images collected by an unmanned aerial vehicle," *Remote Sensing*, vol. 13, no. 5, 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/5/965

[6] P. F. Proença and P. Simoes, "Taco: Trash annotations in context for litter detection," *arXiv preprint arXiv:2003.06975*, 2020.

[7] G. Jocher, "Ultralytics yolov5," 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[8] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[9] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," 2017.

[10] A. Patrizi, G. Gambosi, and F. M. Zanzotto, "Data augmentation using background replacement for automated sorting of littered waste," *Journal of imaging*, vol. 7, no. 8, p. 144, 2021.

[11] S. Majchrowska, A. Mikołajczyk, M. Ferlin, Z. Klawikowska, M. A. Plantykow, A. Kwasigroch, and K. Majek, "Deep learning-based waste detection in natural and urban environments," *Waste Management*, vol. 138, pp. 274–284, 2022.

[12] S. I. Jalal, H. A. Ahmed, and M. H. Ahmed, "Design a robust real-time trash detection system using yolov5 variants," in *2023 IEEE IAS Global Conference on Emerging Technologies (GlobConET)*. IEEE, 2023, pp. 1–6.

[13] D. Das, K. Deb, T. Sayeed, P. K. Dhar, and T. Shimamura, "Outdoor trash detection in natural environment using a deep learning model," *IEEE Access*, 2023.

[14] S. R. Mandhati, N. L. Deshapriya, C. L. Mendis, K. Gunasekara, F. Yrle, A. Chaksan, and S. Sanjeev, "plitterstreet: Street level plastic litter detection and mapping," 2024.

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[16] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, vol. 111, no. 3, pp. 257–276, 2023.