University of Milano-Bicocca
School of Science
Department of Informatics, Systems and Communication
Master of Science in Computer Science

Litter Detection In The Wild Using Efficient Deep Learning Models

Advisor: Prof. Bianco Simone

Co-Advisors: Prof. Schettini Raimondo

Master's Thesis by:
Gaviraghi Elia
Matr. 869493

Academic Year 2023 - 2024

Acknowledgements

Mi concedo la licenza di utilizzare una lingua diversa da quella utilizzata in questa tesi, per potermi esprimere nel profondo con parole sentite. Credo che potrete capire.

Desidero ringraziare tutte le persone che mi hanno accompagnato in questo percorso, accademico e di vita. Rivedo quel ragazzo al primo anno, che iniziò l'università vivendola come un peso e una sofferenza, terminare il suo percorso con nostalgia per un mondo che stava iniziando ad amare. Leggevo i ringraziamenti delle tesi con una certa diffidenza: ora capisco.

Vorrei ringraziare il Prof. Simone Bianco, relatore di questa tesi, il quale mi ha affiancato con pazienza e mi ha permesso l'esplorazione di questo fantastico mondo. Non mi ha trasmesso solo approcci e metodi, ma soprattutto una grande passione. È il più grande risultato, non riassumibile in una tabella, che sia stato ottenuto da questo lavoro.

Ringrazio i professori delle università che partecipano al progetto COBOL: anche la comunità scientifica, con il proprio linguaggio, contribuisce cercando di rendere migliore questo mondo. Ringrazio il dott. Pietro Ghidini per la collaborazione e gli sforzi nel tentativo di costruzione di un dataset di rifiuti, nonostante non siano andati a buon fine. L'Università Bicocca, per il suo spirito materno e la sua umanità, e tutti i Professori che si son dedicati con passione verso Noi studenti.

Sono grato per aver trovato finalmente una qualcosa in questo ambito che non solo mi appassiona. Mi accende. Sentire questa scintilla, dopo anni di indugi, è un riaccendersi di sogni e un grande grido di libertà.

Ringrazio la mia Famiglia, per esser stati vicini in questo zig zag a un figlio che, prima dell'università, ha sempre avuto una vita lineare. Non mi è mai mancato il Vostro supporto e affetto.

Ringrazio i miei Amici e la mia Fidanzata: apprezzo molto, spero un giorno di poter ricambiare questo immenso favore, ricordando in cuor mio che non riuscirò mai ad equiparare il valore del Vostro aiuto, ma vivrò la mia vita provandoci. Ringrazio chi ha iniziato con me questo percorso e chi lo ha finito. *"Chi non c'è resta vivo in un pezzo di te"* canta Fulminacci, e in questo momento di conclusione, vi sento tutti.

Nel film Soul, si direbbe che sono pronto finalmente alla vita. È giunto il momento di saltare.

Summary

Littering presents a significant global challenge, contributing to environmental pollution and endangering wildlife. Automated litter detection is essential for modern waste management, offering efficient monitoring and reducing the reliance on human labor. Citizen science initiatives, where volunteers collect data via smartphones, enhance environmental monitoring but raise privacy concerns, necessitating automated solutions to censor sensitive information like faces and license plates in captured images. To address these needs, lightweight and efficient models, such as YOLO (You Only Look Once), are employed for detection tasks due to their speed and accuracy. YOLO's single-stage approach simultaneously predicts bounding boxes and class probabilities, making it suitable for real-world applications. Despite advancements in the YOLO family, earlier versions like YOLOv5 and YOLOv8 remain highly effective. Recently, YOLO-World introduced open-vocabulary detection, enabling the detection of objects beyond predefined categories, making it adaptable for edge computing and user-generated prompts. This study investigates the development of deep learning models for litter detection, focusing on the strengths and limitations of key datasets. The TACO dataset, recognized for its realistic scenarios and diverse waste types, serves as the primary training source. However, its extensive class count and prevalence of small objects pose challenges for lightweight models. Complementary datasets, such as PlastOPol and UniMiB, provide simpler structures and were used to evaluate model generalization, showing promising results. The UniMiB Trash dataset, featuring categorized waste by size, addresses the limitations of TACO and supports both segmentation and detection tasks. For sensitive information censoring, the WIDER FACE dataset is utilized for face detection, while license plate detection relies on a custom mix of datasets like UFPR-ALPR, addressing the lack of standardized datasets for this task. A new tuning technique of the NMS thresholds with YOLO significantly improved the results on the TACO dataset and all others, leading to the publication of the paper *Efficient Deep Learning Models for Litter Detection in the Wild*, which represents a new state-of-the-art on the TACO dataset. Two-stage prediction techniques is developed to improve the detection of smaller objects with zero-shot ability of YOLO World, such as faces and license plates, using a step-by-step approach inspired by reasoning methods such as Chain of Thought for LLMs. The research demonstrates that a unified YOLO v8 model trained on a combination of datasets can effectively detect litter, faces, and license plates, surpassing individual models trained on the respective datasets in terms of accuracy, speed, and efficiency. These findings validate the potential of unified models for diverse object detection tasks in real-world scenarios, contributing to advancements in environmental monitoring and automated privacy protection.

Contents

Chapter 1

Introduction

1.1 Problem Definition

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.

Litter detection in the wild is a challenging task for current computer vision and Deep Learning models. Litter is difficult to detect due to certain intrinsic properties. Any object can be considered litter, meaning that the items to be detected can have a wide variety of shapes and colors. Additionally, degradation may occur after abandonment, causing the object to differ significantly from its original state in ways that are impossible to replicate artificially. Abandoned waste in nature can also be occluded by plants and weeds that grow over time, creating significant occlusion challenges for deep learning models. Moreover, up until now, research in the field has focused mainly on relatively simple scenarios, with only a few small or medium-sized litter items. In reality, however, much more complex situations arise, where accumulated waste can form actual illegal dumps. These scenarios are not only difficult to identify with current methods, but they also present challenges in defining the problem itself, from annotating these images to determining the appropriate metrics to use. Adding to the challenge is the fact that this problem needs to be solved not by large models running in multi-GPU data centers, but potentially on devices with limited computational capacity, such as smartphones, making it even more difficult to achieve good results. In addition to detecting abandoned waste in nature, it is also necessary to censor sensitive information that users may include in their images, either intentionally or unintentionally, such as people's faces or vehicle license plates.

1.2 Datasets overview

This research utilizes several dataset. For litter detection, Trash Annotation Context (TACO)[26], PlastOPol [7] and the collected UniMiB Trash are used. TACO contains 1,500 high-resolution images with a total of 4,784 annotations, which are divided into 60 classes an grouped into 28 supercategories. PlastOPol comprises 2,418 images with 5,300 litter instances under a single Litter class. The UniMiB Trash dataset, collected by the authors of this thesis, addresses the previous datasets. It contains approximately 1,200 images with 6,289 instances, formatted for both segmentation and detection tasks, with five classes based on the size of litter instances: micro, small, medium, large and cluster. For face detection, the WIDER FACE[41] dataset is among the best options available. This extensive dataset contains 32,203 images and 393,703 labeled faces, selected from the publicly available WIDER dataset. For license plate detection tasks, a mixed approach was adopted, utilizing the UFPR-ALPR[21], License Plates[32], and Car License Plate Detection[5] datasets. The UFPR-ALPR dataset includes 4,500 images extracted from 150 videos in urban traffic. Since the images extracted from the same video are too similar, only 1 image per video was retained, resulting in a total of only 150 images. License Plates Dataset, a subset of the Open Images Dataset[20], contains approximately 350 images with an official split. The Car License Plate Detection dataset includes 433 low-resolution images with 471 instances.

1.3 Contributions and Motivations

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).

COBOL[2] is a National project funded by the Italian MUR started in December 2023 with the aim of defining an advanced framework for managing the waste disposal process. COBOL involves three universities (the University of Milano - Bicocca, which coordinates the project, the Polytechnic of Milan, and the Gran Sasso Science Institute) and interacts with multiple Italian municipalities interested to experiment with the developed services. COBOL works on the definition of a decentralized data-processing architecture that exploits gamification to involve all the relevant stakeholders in the waste disposal process; model-driven engineering, to enable the creation of highly flexible and executable waste disposal models; computer vision, to help identify, sort and categorize the reported waste effectively; federated learning, to enable the sharing of knowledge among different communities involved in waste management without breaking privacy requirements; and self-adaptation, to ensure the capability to face unexpected events. Early results show that the federated learning architecture can be effectively used to collect reports and computer vision techniques applied to real-life datasets can be used to semi-automate the littering detection process.

This thesis is part of a broader project involving several universities aimed at proposing innovative methods to improve the management of abandoned waste. This includes exploring software architecture and analyzing federated learning techniques to determine whether user privacy protection and the training of deep learning models can coexist effectively. The personal focus is on the topic of deep learning, specifically to understand whether the task is feasible with current models and data, to identify the real-world problems and dilemmas that need to be addressed, and to explore potential solutions. This

work emphasizes the development of extremely lightweight deep learning models that can be run on smartphones.

On a personal level, I feel that the opportunity to spend my time and energy in order to contribute to making this world a better place for all is a unique opportunity to connect my humanity with my professional field. Studies like this, although they may not change the world (immediately), can demonstrate that the emerging field of artificial intelligence represents a unique opportunity for humanity. There exists an ethical framework and a willingness to use AI to contribute positively to society, rather than viewing it solely as an enemy. In this context, the models developed are not intended to replace human roles in the identification and management of abandoned waste, especially considering that, given certain states of degradation, human resources apparently do not really exist or are not efficient.

1.4 Thesis Structure Overview

This thesis is structured into several chapters, each addressing a distinct aspect of the research undertaken. The organization aims to guide the reader through the foundational concepts, methodologies, experimental results, and conclusions, culminating in recommendations for future research.

Chapter 2 delves into the state-of-the-art models and techniques critical to this thesis. The first version of You Only Look Once (YOLO), a single-stage object detector, is described. This is followed by the introduction of YOLO v2, v3, v4, v5 and v8. The chapter also explores YOLO World, the first Open-Vocabulary Detector (OVD) based on YOLO and with potentially real-time performance.

In Chapter 3, attention shifts to the datasets utilized in this study. First, the three main datasets for litter detection are described: TACO, PlastOPol and the dataset created by the authors UniMiB Trash. Next, the datasets used for censoring sensitive information are described: Wider Face for faces, and UFPR-ALPR, License Plates Dataset and Car License Plate Detection. Other licence plate datasets analysed, but not included in the mixed dataset created with the reasons attached, are also described.

Chapter 4, named methods, describes two experimental methodologies developed in this thesis. First, the experimental tuning of the Non-Maximum Suppresion algorithm is described, initially applied to the TACO dataset and then used in all the other experiments conducted. Next, an entire section is devoted to YOLO World, in particular, improvements made by improving the input prompt are introduced and a new prediction mode realised, called Two-Stage, is described.

Next, Chapter 5 presents the results obtained on various datasets, both with and without the developed methodologies. It begins with the results on the TACO-1 task, including details on the training procedures and setup. The obtained models are then tested on the PlastOPol and UniMiB datasets, also using a single class. The UniMiB dataset is further utilized for a new task involving four classes. The chapter concludes with the results on the datasets related to faces and license plates. These results, which involve developing a model for each class, are then compared with the individual models trained on all three classes simultaneously.

The thesis culminates in Chapter 6, which summarises the findings, presents the conclusions drawn from the research, and proposes directions for future investigation.

Chapter 2

State of the Art

2.1 Literature Review

This section meticulously evaluates the existing the state-of-art regarding the models used, particularly the history of the YOLO model family used to achieve the object detection task. The examination begins with a focus on the first YOLO model built, describing the architecture and general ideas of this revolutionary detector.

Then, later versions of YOLO as the time of this writing are described, based on the differences from the previous model: YOLO v2, v3, v4, v5 and v8. There are many versions of YOLO, such as Gold-YOLO [37], YOLOX [10], YOLO v6 [23] and v7 [38]. Some of these models are marginal and have not very success; moreover, since there is no competent body or standard, the name of YOLO models are defined by the researchers themselves, thus the sequence is only temporal but does not indicate a necessary advancement substantiated by a third party.

2.2 You Only Look Once

You Only Look Once (YOLO), or YOLO v1, is an innovative and fast object detection model that addresses the problem as a single spatial regression from the input image to the bounding boxes and class probabilities. Introduced by Joseph Redmon et al [30] in 2016, YOLO offers significant speed improvements over traditional detection methods.

YOLO uses a single convolutional neural network (CNN) to simultaneously predicts multiple bounding boxes and class probabilities for those boxes.

The core principle proposed by YOLO-v1 was the imposing the division of the input image into a grid $S \times S$ (default $= 7 \times 7$) and for each grid cell, the algorithm predicts bounding boxes and confidence of prediction. In the case of the center of the object of interest falling into one of the grid cells, that particular grid cell would be responsible for the detection of that object: this permitted other cells to disregard that object in the case of multiple appearances. For implementation of object detection, each grid cell would predict B bounding boxes along with the dimensions and confidence scores. The confidence score was indicative of the absence or presence of an object within the bounding box. Therefore, the confidence score can be expressed as

$$\text{conf\_score} = p(\text{object}) \cdot \text{IoU}_{\text{pred}}^{\text{truth}}$$

where $p(object)$ signified the probability of the object being present, with a range of $[0, 1]$.

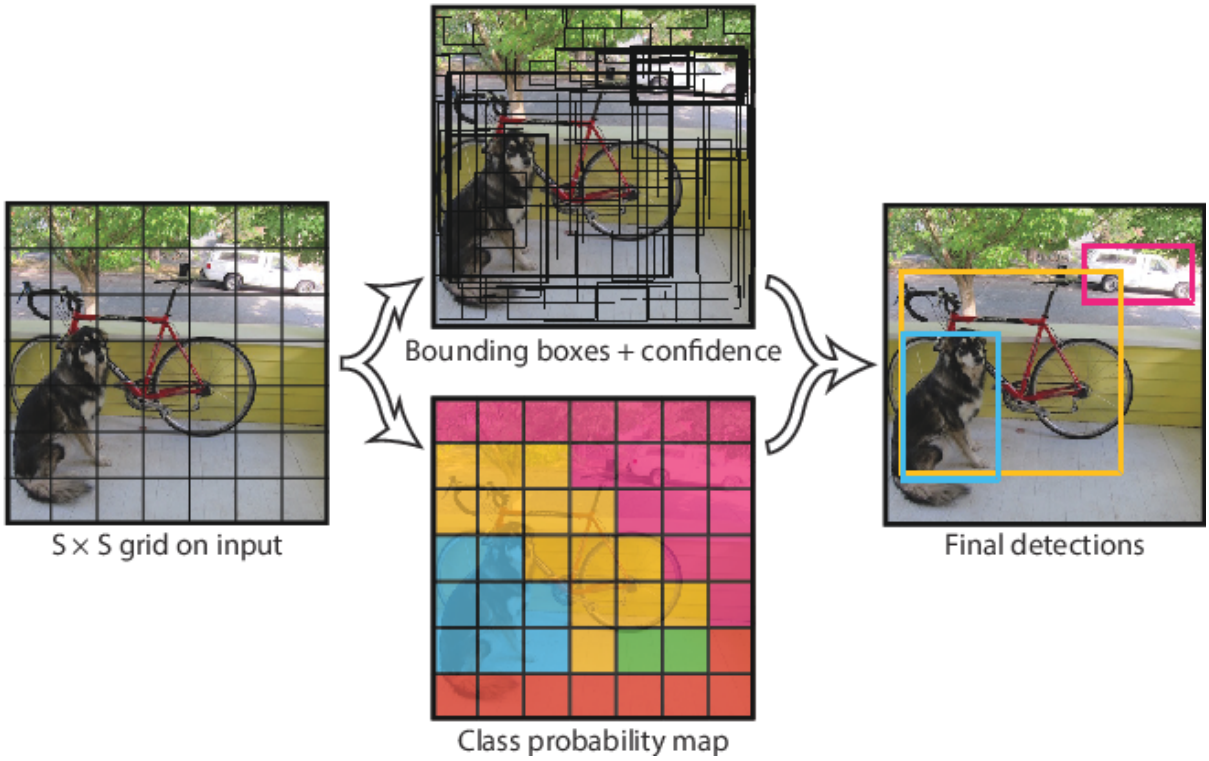

Figure 2.1: YOLO addresses the problem as a single spatial regression. Image taken from [30]

Each bounding box is composed of 5 predictions: x, y, w, h and *confidence score*. The coordinates (x, y) denote the center of the box in relation to the grid cell boundaries. The width and height are predicted in relation to the entire image. Lastly, the confidence prediction signifies the Intersection over Union (IOU) between the predicted box and any actual ground truth box.

Architecture Overview

YOLO model takes a fixed-size input image (448×448) and passes it through a convolutional neural network (CNN) to extract features and produce predictions of bounding boxes and classes.

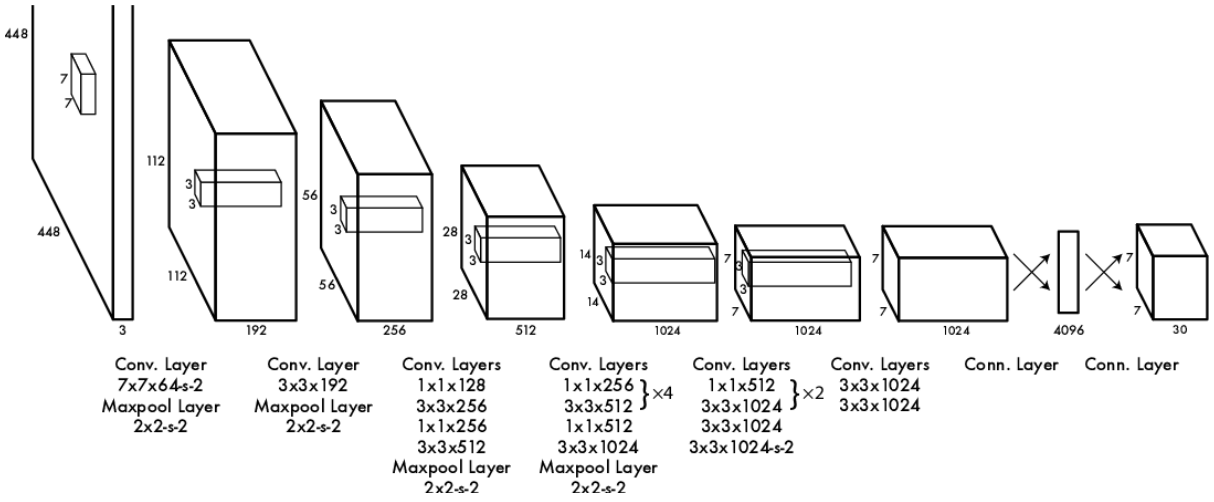

Figure 2.2: Backbone architecture of YOLO, image taken from [30]

Pre-trained on ImageNet, this network architecture is inspired by the GoogLeNet model for image classification where, instead of the initial modules used by GoogLeNet, the model use a 1×1 reduction layers followed by 3×3 convolutional levels to reduce the resultant feature space from the preceding layers. The architecture consists of 24 convolutional layers followed by 2 fully connected layers: as can be seen in Figure 2.2 the initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates.

As explained earlier, the input image is split into $s \times s$ grid cells, with each cell predicting B bounding boxes, each containing five parameters and sharing prediction probabilities of classes (C). Therefore, the parameter output would take the following form:

$$s \times s \times (B \cdot 5 + C$$

Considering the example of YOLO network with default parameter about the split cells, and each cell bounding box prediction set to 2 and evaluating the benchmark COCO dataset consisting of 80 classes, the parameter output would be given as expressed in

$$7 \times 7 \times (2 \cdot 5 + 80)$$

NMS

To address multiple bounding boxes containing no object or the same object, YOLO use the Non-Maximum Suppression (NMS) [15] algorithm. By defining a threshold value for NMS, all overlapping predicted bounding boxes with an IoU lower than the defined NMS value are eliminated.

YOLO vs R-CNN

R-CNN[13] and its variants use region proposals instead of sliding windows to find objects in images. Each stage of a complex pipeline must be precisely tuned independently:

- Selective Search generates potential bounding boxes
- A convolutional neural network (CNN) extracts features.

- An SVM scores the boxes
- Linear model adjusts the bounding boxes
- The Non-Maximum Suppression eliminates duplicate detections

As this execution pipeline is extremely slow, the Fast[12] and Faster[31] R-CNN were introduced to focus on speeding up the R-CNN framework by sharing computation and using neural networks to propose regions instead of Selective Search. While they offer speed and accuracy improvements over R-CNN, both still fall short of real-time performance.

YOLO shares some similarities with R-CNN, but it's extremely faster compared to the previous: each grid cell proposes potential bounding boxes and scores those boxes using convolutional features. However, the system puts spatial constraints on the grid cell proposals which helps mitigate multiple detections of the same object. System also proposes far fewer bounding boxes compared to Selective Search. Finally, system combines these individual components into a single, jointly optimized model.

2.3 YOLO v2

YOLO v2, also called YOLO 9000, was introduced by [28] immediately after YOLO v1. The focus of this version is improving recall and localization while maintaining classification accuracy. YOLO v2 adds three important changes from the previous version.

Batch Normalization Layers

YOLO v2 apply a batch normalization layer to all convolutional layers. Introducing batch normalization leads to substantial improvements in convergence, simultaneously obviating the need for other regularization techniques. By adding these layers on all of the convolutional layers in YOLO, the authors get more than 2% improvement in terms of mean Average Precision (mAP) on Pascal VOC 2007[9]; furthermore, batch normalization contributes to model regularization, allowing to eliminate dropout without risking overfitting.

New Darknet Backbone

YOLO v2 used a custom deep architecture Darknet-19, which has 19 convolutional layers, 5 max pooling layers and a softmax layer for classification objects. This backbone achieved 91.2% top-5 accuracy on ImageNet: it's better than VGG (90%) and the first YOLO v1 network(88%).

High Resolution Classifier

State-of-the-art detection methods rely on classifiers pre-trained on ImageNet with input images smaller than 256×256. In the original YOLO [30], the authors train the network with 224×224 and then increase the resolution to 448×448 for detection.

For YOLOv2 the authors first fine tune the classification network at the full 448×448 resolution for 10 epochs on ImageNet - 416×416 with the addition of anchor boxes. This gives the network time to adjust its filters to work better on higher resolution input; then

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Figure 2.3: The backbone of YOLO v2, darknet-19. Image taken from [28]

fine tune the resulting network on detection: high resolution classification network gives leads to increase performance by 4% in terms of mAP.

Fine-Grained Features

While a a 13×13 feature map is sufficient for large objects, it may benefit from finer grained features for localizing smaller objects, where YOLO struggles the most. Faster R-CNN and SSD both run their proposal networks at various feature maps in the network to get a range of resolutions. YOLO v2 take a different approach, simply adding a passthrough layer that concatenates the higher resolution features with the low resolution features by stacking adjacent features into different channels instead of spatial locations, that turns the $26 \times 26 \times 512$ feature map into a $13 \times 13 \times 2048$ feature map, which can be concatenated with the original features.

Convolutional With Anchor Boxes

YOLO directly predicts the coordinates of bounding boxes using fully connected layers on top of the convolutional feature extractor. To enhance YOLO, the authors eliminating the fully connected layers and employing anchor boxes for bounding box prediction, as use in Faster R-CNN[31]. First, the authors remove the one pooling layer to make the output of the network's convolutional layers higher resolution. Secondly, because large object tend to occupy the center of the image, it is good to have a single location right at the center to predict these objects, instead of four locations that are all nearby: YOLO's convolutional layers downsample the image by a factor of 32 so by using an input image of 416 we get an output feature map of 13×13. Using anchor boxes a small decrease in accuracy is observable, but recall increases significantly: on Pascal VOC, without anchor boxes the model gets 69.5 mAP with a recall of 81%; using anchor boxes the model gets 69.2 mAP, but with a recall of 88%.

Direct location prediction

Introducing anchor boxes with YOLO encounters a several issue, one of which is model instability, especially during early iterations. Most of the instability comes from predicting the (x, y) location for the box. In region proposal networks the network predicts values t_x and t_y and the (x, y) center coordinates are calculated as:

$$x = (t_x \cdot w_a) - x_a$$

$$y = (t_x \cdot h_a) - y_a$$

This formulation is unconstrained so any anchor box can end up at any point in the image, regardless of what location predicted the box. With random initialization the model takes a long time to stabilize to predicting sensible offsets. Instead of predicting offsets, this method follow the approach of YOLO and predict location coordinates relative to the location of the grid cell, using a logistic activation to constrain the network's predictions to fall in the range $[0, 1]$. Constraining the location prediction, the authors obtain an easier parametrization, making the network more stable. Using dimension clusters along with directly predicting the bounding box center location improves YOLO by almost 5% over the version with anchor boxes.

Multi-Scale Training

The authors want YOLO v2 to be robust to running on images of different sizes, so they train this into the model, we change the network every few iterations. Every 10 batches during the training, the network randomly chooses a new image dimension size that is multiple of 32, forcing the network to learn to predict well across a variety of input dimensions. At 288×288 it runs at more than 90 FPS on NVIDIA Titan X with mAP almost as good as Fast R-CNN (69%), while at higher resolution YOLOv2 was a state-of-the-art detector with 78.6 mAP on Pascal VOC 2007.

2.4 YOLO v3

YOLO v3 was introduced by [29] 2 years after the release of YOLO. It introduces a deeper backbone, a new prediction across scales and other new features to improve results,

especially for small objects and more complex detection scenarios, while retaining the advantage of real-time speed, a peculiarities of the YOLO family. YOLO v3 is a little bigger than the previous, but it's more accurate: the authors particularly compare it with RetinaNet-50 and RetinaNet-101 on COCO[24] dataset, as can be seen from the Figure 2.4

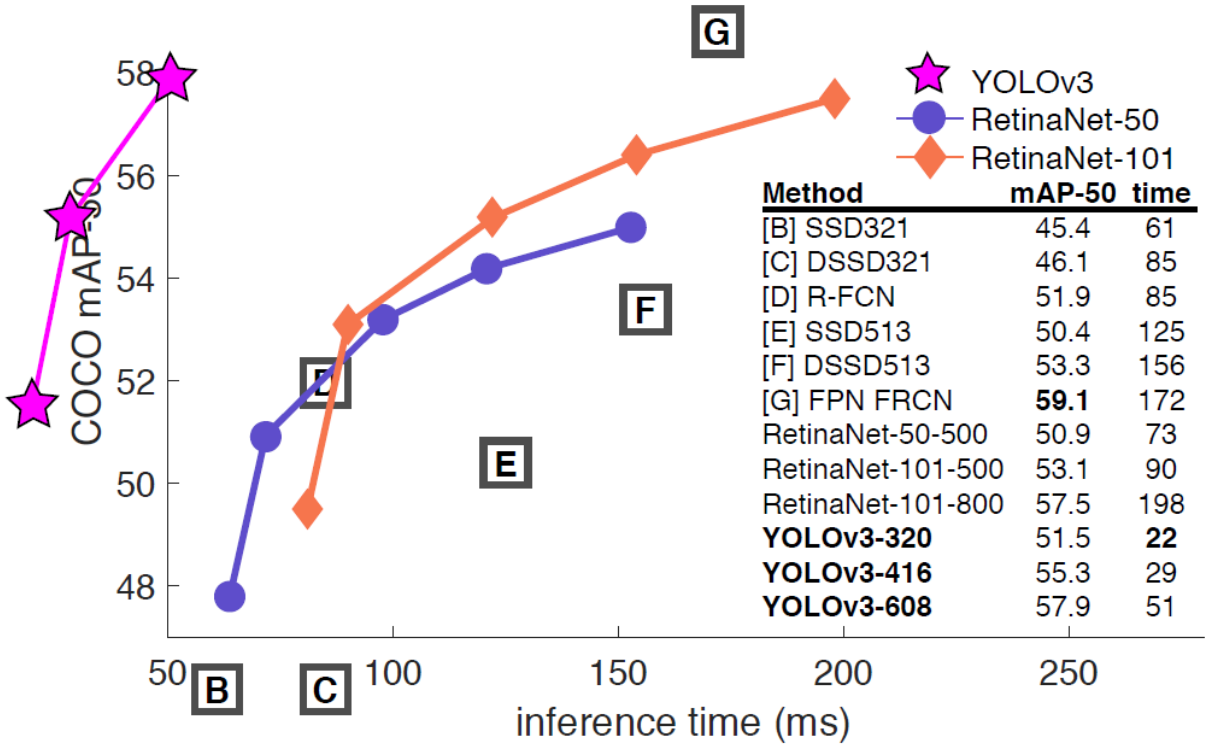

Figure 2.4: Performance of YOLO v3 at three different scales on COCO dataset. Image taken from [29]

Darknet-53

YOLOv3 uses new backbone to performing feature extraction. The new network is a hybrid approach between the network used in YOLOv2 (Darknet-19) and the residual network, so it has some shortcut connections. This architecture has successive 3×3 and 1×1 convolutional layers, but now has some shortcut connections as well and is significantly larger: went from 19 convolutional layers of YOLO v2 to 53 convolutional layers in YOLO v3, calling the backbone Darknet-53. Details of architecture can be seen in Figure 2.5.

Prediction across scales

This is the most salient features: unlike YOLO v1 and YOLO v2 which predict the output at the last layer, YOLO v3 predicts boxes at 3 different scales, similar to the FPN (Feature Pyramid Networks) where 3 predictions are made for every location the input image and features are extracted from each prediction. The scales are precisely given by downsampling the dimensions of the input image by 32, 16 and 8 respectively. At

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 2.5: Backbone of YOLO v3, the Darknet-53. Image taken from [29]

each scale, YOLO v3 uses 3 anchor boxes and predicts 3 boxes for any grid cell; each object is still only assigned to one grid cell in one detection tensor. This method allows YOLO v3 to get more meaningful semantic information from the upsampled features and finer-grained information from the earlier feature map.

Multi labels prediction

YOLO v3 performs multilabel classification for objects detected in images replacing softmax function with independent logistic classifiers.

In some datasets like the Open Image Dataset, there are many overlapping labels, because an object may have multi labels, i.e. an object can be labeled as a mammal and as cat: if instead we used the a softmax, it would impose the assumption that each box has exactly one class which is often not the case. For this reason, YOLO v3 doesn't use a softmax function but uses a simple independent logistic classifiers for any class: in this

way an object can be detected as a mammal an as a cat at the same time.

2.5 YOLO v4

The fourth generation of YOLO implemented by [3] is presented: the main goal of the new authors is designing a fast operating speed of an object detector in production systems and optimization for parallel computations, so they created a new SOTA model as shown in Figure 2.6

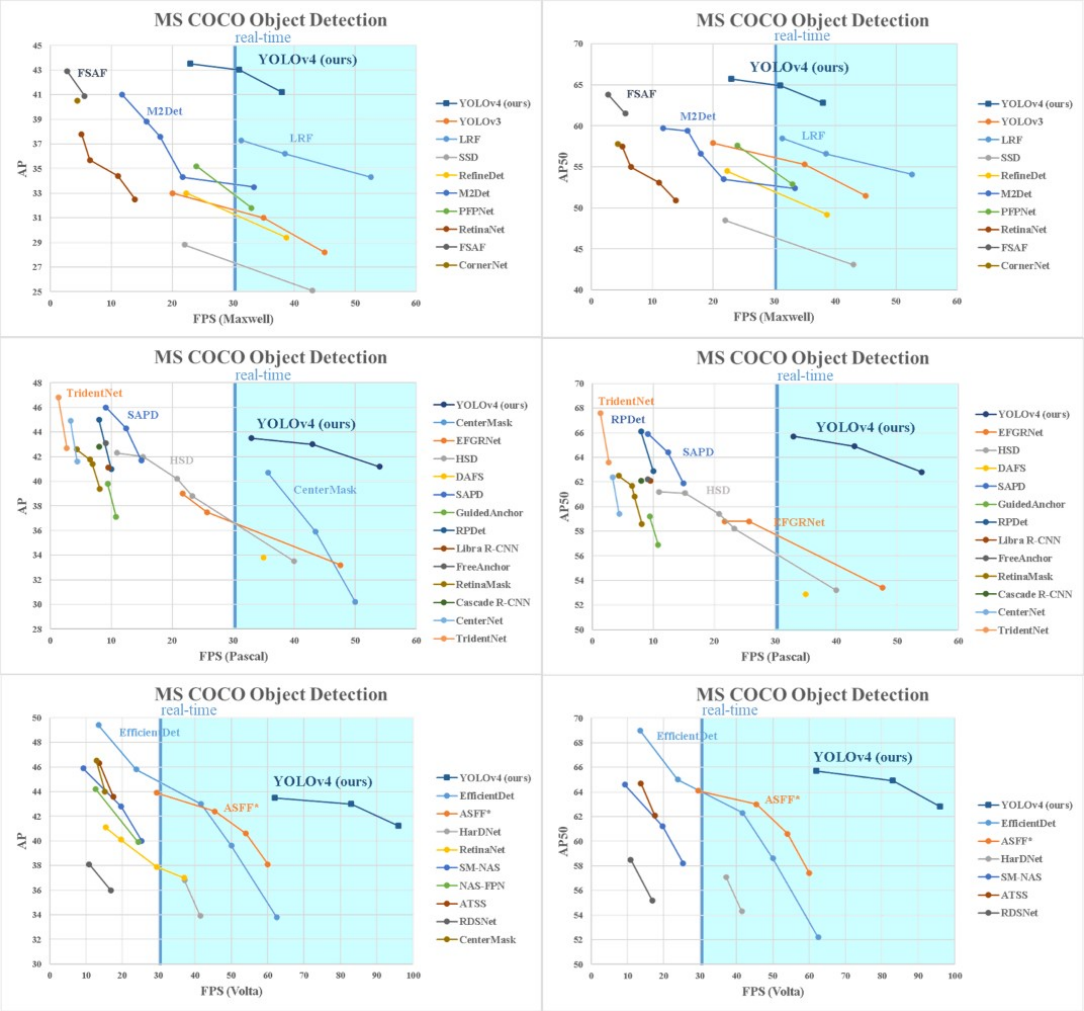

Figure 2.6: YOLO v4 performance on COCO dataset with different GPU architecture: the model remains a real-time model for more GPUs compared to several famous models. Image taken from [3]

CSPDarknet53 as backbone

The objective of authors is to find the optimal balance among the input network resolution, the convolutional layer number, the parameter number and the number of layer outputs (filters). The authors considered the following backbones for the YOLOv4 object detector: CSPResNext50, CSPDarknet53 and EfficientNet-B3. The CSPResNext50 and the CSPDarknet53 are both based on DenseNet, which was designed to connect layers in

convolutional neural networks to alleviate the vanishing gradient problem, encourage the network to reuse features and reduce the number of network parameters. The idea with the CSPResNext50 and CSPDarknet53 is to remove computational bottlenecks in the DenseNet and improve learning by passing on an unedited version of the feature map. Instead EfficientNet[35] was designed by Google Brain to primarily study the scaling problem of convolutional neural networks. This network outperforms the other networks of comparable size on image classification, but a reference model which is optimal for classification isn't always better for a detector: in contrast to the classifier, an object detector requires Higher input network size (resolution) to detect multiple small-sized objects, more layers for a higher receptive field to cover the increased size of input network and more parameters to increase the capacity of detect multiple objects which have different size.

After several experiments, the authors choose the CSPDarknet53 as backbone witch SPP additional module, PANet path-aggregation neck and YOLOv3 head to form the YOLO v4 architecture, which contains 29 convolutional layers 3×3, a 725×725 receptive field and 27.6 M parameters

Bag of Freebies

Bag of Freebies (BoF) refers to methods that improve the training accuracy of YOLOv4 without increasing the cost of inference. Most of the Bag of Freebies have to do with data augmentation (see Figure 2.7): the most important are SAT, Mosaic and CutMix; they are used to expand the training set and build a model capable a model that can generalise better.

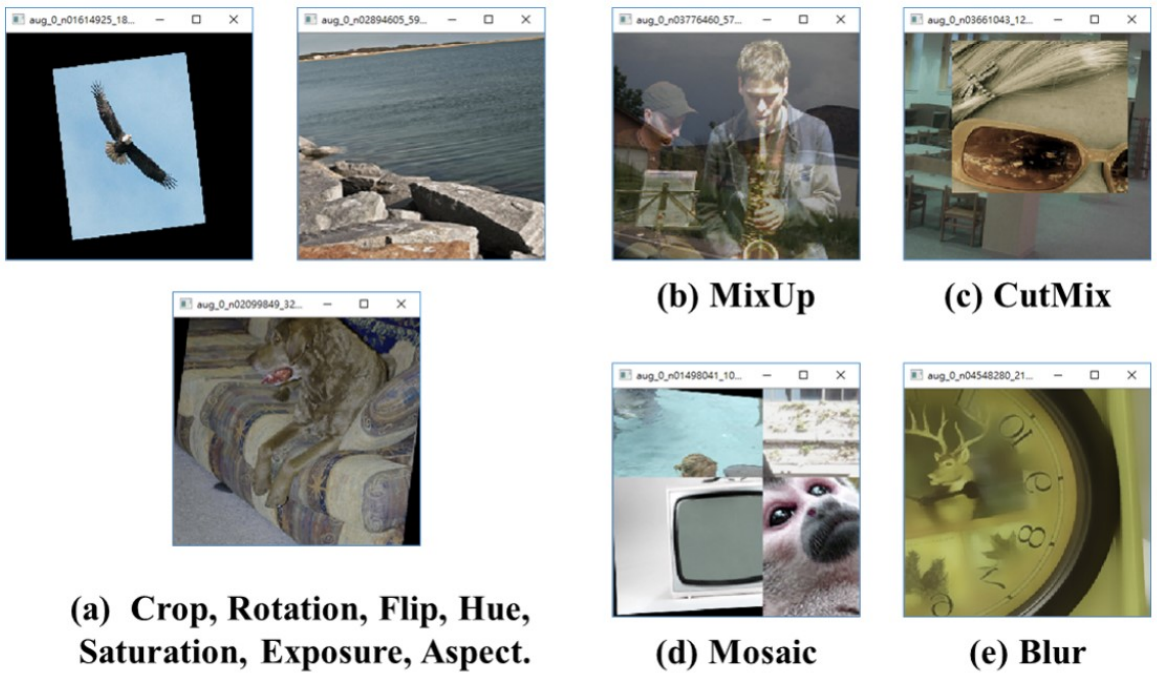

Figure 2.7: Example of some techniques augmentation evaluated for YOLO v4. Images taken from [3]

Self-Adversarial Training (SAT) is a new data augmentation technique that operates in 2 forward backward stages:

1. Adversarial Image Generation.
   In this first stage, the Neural Networks slightly modifies the original image instead of the network weights. The modifications are done in such a way that the resulting images are adversarial examples that could potentially fool the model by manipulating the pixel values in the image based on the gradients obtained from the model's predictions. In this way the neural network executes an adversarial attack on itself, altering the original image to create the deception that there is no desired object on the image.

2. Training on Adversarial Examples
   In this second stage, the neural network is trained to detect an object on this modified image. In this way the model becomes more robust and improves its ability to generalize to real-world scenarios.

Mosaic represents a new data augmentation method that mixes 4 training images with different context: it takes four images and stitches them together to form a single mosaic image. This technique effectively increases the batch size, leading to more stable training, and provides more context for objects, improving the model's ability to detect objects in various scales and locations by providing more complex scenarios.

CutMix techniques mixes only 2 input images: this method cuts a rectangular region from one image and pastes it onto another image; the labels (bounding boxes) are also adjusted according to the regions being mixed. CutMix provides regularization by mixing parts of different images, which helps prevent overfitting and improve training efficiency and robustness by presenting the model with more varied and unpredictable scenarios.

YOLO v4 use DropBlock[11] as regularization method. In DropBlock, sections of the image are hidden from the first layer: this technique force the network to learn features that it may not otherwise rely upon. Researchers who published DropBlock have compared their method with other methods in detail, and their regularization method has won a lot. Therefore, the authors of YOLO v4 didn't hesitate to implement DropBlock for their architecture.

Bag of Special

Bag of Special is a set of features that add marginal increases to inference time, but significantly increase performance improves performance.

The authors experiment with various activation functions: ReLU, leaky-ReLU, parametric-ReLU, ReLU6, SELU, Swish, and Mish. Activation functions transform features as they flow through the network. Since Parametric ReLU and SELU are more difficult to train, and ReLU6 is specifically designed for quantization network, the authors therefore remove the above activation functions from the candidate list.

With traditional activation functions like ReLU, it can be difficult to get the network to push feature creations towards their optimal point. So the research has been done to produce functions that marginally improve this process. Mish is an activation function designed to push signals to the left and right. At the end, Mish activation function was selected.

2.6 YOLO v5

YOLOv5, developed in PyTorch by Ultralytics in 2020 [36]. The documentation is weak: no papers are available like previous versions. YOLO-v5 similar to YOLO-v4, with respect to contributions, focus on the conglomeration and refinement of various computer vision techniques for enhancing performance. A notable mention is that YOLO-v5 was the first native release of architectures belonging to the YOLO clan, to be written in PyTorch rather than Darknet. Although Darknet is considered as a flexible low-level research framework, it was not purpose built for production environments with a significantly smaller number of subscribers due to configurability challenges. PyTorch, on the other hand, provided an established eco-system, with a wider subscription base among the computer vision community and provided the supporting infrastructure for facilitating mobile device deployment.

YOLOv5 was released with five different sizes: nano (n), small (s), medium (m), large (l) and extra large (x): the largest version of the YOLO v5x model and the smallest YOLO v5n do not differ in layers: the difference between them as well as between other versions is in the scaling multipliers of the width and depth of the network. In this research we worked with the lightest models, the *nano* and *small* respectively.

The input sizes of YOLO v5 is variable, commonly fixed to 640×640, allowing for more flexibility. The backbone is designed using the New CSP-Darknet53 structure, a modification of the Darknet architecture used in previous versions [3]. The head of YOLO v5 is responsible for generating the final output, so YOLO v5 uses the same of YOLO v3 [29]. The connection between the backbone and the head is achieved via the neck: YOLO v5 uses the SPPF and New CSP-PAN structures.

Cross Stage Partial Network

YOLO v5 incorporates both residual and dense blocks to facilitate information flow to the deepest layers and to address the vanishing gradient issue. However, one of the challenges of using dense and residual blocks is the issue of redundant gradients. CSPNet addresses this issue by truncating the gradient flow, maintaining the benefits of DenseNet's feature reuse capabilities, and reducing the surplus of redundant gradient information by truncating the gradient flow. YOLOv5 utilizes the CSPNet strategy to divide the base layer's feature map into two sections, which are then combined through a cross-stage hierarchy. This strategy offers significant benefits to YOLOv5, as it reduces the number of parameters and significantly decreases computation (FLOPS), leading to an increase in inference speed, a critical factor in real-time object detection models.

Training strategies

YOLOv5 applies several sophisticated training strategies to enhance the model's performance. In multiscale training, as the previous versions, use a range of 0.5 to 1.5 to randomly rescaled the input images during the training process.

Data Augmentation Techniques

YOLOv5 employs various data augmentation techniques to improve the model's ability to generalize and reduce overfitting. These techniques include Mosaic, Copy-Paste, Random Affine, MixUp, Albumentations, HSV and Random Horizontal Flip. Mosaic

combines four training images into one in ways that encourage object detection models to better handle various object scales and translations. MixUp creates composite images by taking a linear combination of two images and their associated labels. Copy-Paste copies random patches from an image and pastes them onto another randomly chosen image, effectively generating a new training sample. Albumentations [4] is a powerful library for image augmenting that supports a wide variety of augmentation techniques. Random Affine includes random rotation, scaling, translation, and shearing of the images. HSV augmentation applies some random changes to the Hue, Saturation, and Value of the images.

2.7 YOLO v8

YOLO v8 [18] was launched by Ultralytics on January 10th, 2023. Building on the foundational YOLOv5, YOLOv8 introduces several key improvements as shown in Figure 2.8. Like v5 and v6, YOLOv8 has no official paper released.

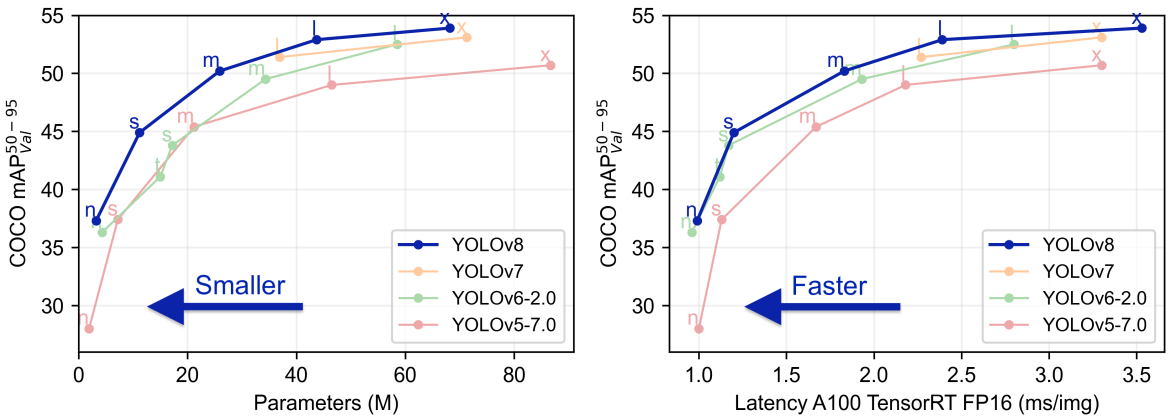

Figure 2.8: YOLO v8 performance on COCO [24] dataset. The performance in terms of mAP50-95/Number of parameters **(left)** and also in terms of mAP50-95/Latency calculated on a NVIDIA A100 GPU with FP16 precision **(right)**

A key update in YOLOv8 is the move to anchorless detection, moving away from the anchor-box techniques of YOLO v5 [36]. This modification allows the centres of objects to be predicted directly, simplifying the Non-Maximum Suppression (NMS) process and alleviating the problems associated with anchor boxes, such as the lack of generalisation and the difficulty in handling irregular shapes.

In YOLOv8's convolutional design, the initial 6×6 convolution in the stem is replaced with a more efficient 3×3 convolution. The core building block is also updated, replacing the C3 module with C2f. The authors also delete two convolutional layers (the number 10 and 14 respectively in the YOLOv5 configuration). Additionally, features in the model's neck are concatenated directly without requiring matching channel dimensions, reducing both the parameter count and tensor size. A new feature called Spatial Pyramid Pooling Feature (SPPF) is introduced, improving the model's capability to handle objects of various scales.

Similar to YOLOv5, YOLOv8 is also available in five different versions, each of which is designed to balance detection accuracy and computational efficiency.

2.8 YOLO World

Prevalent object detection research concentrates on fixed vocabulary (close-set) detection, in which object detectors are trained on datasets with pre-defined categories, usually on COCO [24] dataset, limiting their ability to detect objects beyond these predefined classes without retraining on custom datasets.

Open-vocabulary object detection (OVD) has emerged as a new trend for modern object detection, which aims to detect objects beyond the predefined categories: models like CLIP, Grounding DINO and F-VLM have demonstrated the potential of vision-language modeling in object detection task.

Early works follow the standard OVD setting by training detectors on the base classes and evaluating the novel classes. Nevertheless, this open-vocabulary setting can evaluate the capability of detectors to detect and recognize novel objects, it is still limited for open scenarios and lacks generalization ability to other domains due to training on the limited dataset and vocabulary.

YOLO-World [6] addresses limitations of traditional object detection methods by enabling open-vocabulary detection beyond fixed categories, offering adaptability to new tasks, reducing computational burden, and simplifying deployment on edge devices.

YOLO-World retains the real-time performance characteristic of the YOLO architecture: this is crucial for applications where timely detection of objects is required, such as in autonomous vehicles or surveillance systems.

Model Architecture

YOLO-World's architecture consists of three key elements: YOLO detector, text encoder and re-parameterizable vision-language path aggregation network (RepVL-PAN). The overall architecture of YOLO-World is shown in Figure 2.10.

The YOLO detector, built on YOLO v8 [18], extracts multi-scale features from the input image. At the same time, the text encoder, pre-trained by CLIP, processes the input text and converts it into text embeddings. The CLIP text encoder offers superior visual-semantic capabilities for linking visual objects with text compared to text-only language encoders. When the input text is a caption or referring expression, YOLO World uses a simple n-gram algorithm to extract noun phrases, which are then fed into the text encoder. The RepVL-PAN is subsequently used to enhance the representations of both text and image by facilitating cross-modality fusion between the image features and text embeddings.

The fusion of image features and text embeddings is achieved using a Text-guided Cross Stage Partial Layer (T-CSPLayer) and Image-Pooling Attention. The T-CSPLayer builds upon the C2f layer from the YOLOv8 architecture by incorporating text guidance into the multi-scale image features. This is done through a Max Sigmoid Attention Block, which calculates attention weights based on the interaction between text guidance and the image's spatial features. These weights are applied to adjust the feature maps, allowing the network to focus more on areas that are relevant to the text descriptions. Image-Pooling Attention further refines the text embeddings by integrating visual context, using max pooling on multi-scale image features to distill them into 27 patch tokens that capture

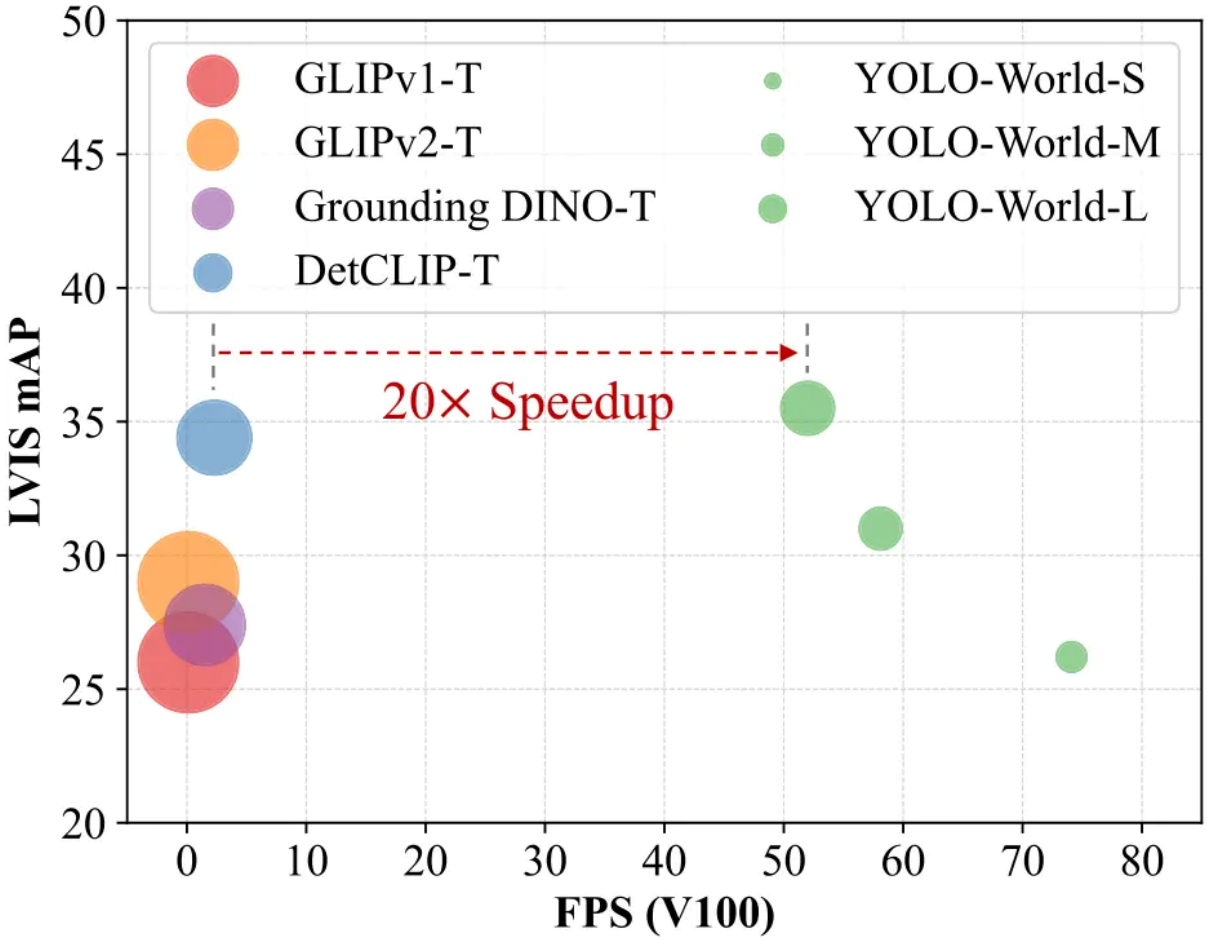

Figure 2.9: Comparison between YOLO World models and the other recent open-vocabulary methods in terms of speed and accuracy. Image taken from [6]

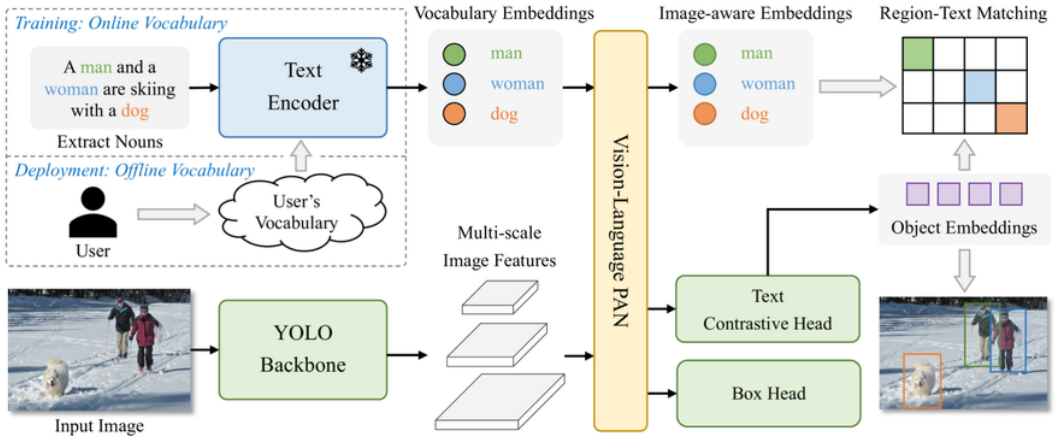

Figure 2.10: The architecture of YOLO-World introduces a novel approach compared to traditional YOLO detectors by incorporating text as part of its input, enabling open-vocabulary detection. The Text Encoder processes the input text, generating embeddings that represent the textual information. Simultaneously, the Image Encoder extracts multi-scale features from the input image. The proposed RepVL-PAN module then performs multi-level cross-modal fusion, integrating both image and text features. In the final stage, YOLO-World predicts regressed bounding boxes and generates object embeddings, which are matched to the categories or nouns provided in the input text.

essential regional information. These tokens are then processed through a mechanism involving queries derived from text embeddings, along with keys and values from image patches, to compute scaled dot-product attention weights.

Online and Offline vocabulary

YOLO World uses online vocabulary during training, to construct an online vocabulary T for each mosaic sample containing 4 images. Specifically, the model sample all positive nouns involved in the mosaic images and randomly sample some negative nouns from the corresponding dataset. The vocabulary for each mosaic sample contains at most M nouns, and M is set to 80 as default. In the inference part, YOLO World uses a different strategy, choosing an Offline Vocabulary. At the inference stage, this model present a prompt-then-detect strategy with an offline vocabulary for further efficiency. Instead of using an Online vocabulary (i.e., one that can be dynamically updated), the user generates a series of prompts (phrases or descriptions) according to their needs. These prompts are then encoded into an offline vocabulary, which is a fixed set of object descriptions that the model can detect. Once encoded, this vocabulary can be "re-parameterized" as the model weights, allowing for further optimization and acceleration during inference. This strategy allows for avoiding computation for each input and provides the flexibility to adjust the vocabulary as needed.

Chapter 3

Datasets

In this section, the datasets used and collected in this work are presented. First, the state-of-the-art datasets related to waste, TACO and PlastOPol, selected after a thorough search, are introduced. Then, the collected dataset and the related challenges are described in detail, from the image collection process to the final considerations regarding the release of the dataset. Finally, the datasets used to train the models for censoring faces and license plates are presented.

3.1 TACO

The TACO dataset [26] contains 1,500 high-resolution images with a total of 4,784 annotations. The image resolutions range from 842×474 to $6,000 \times 4,000$ pixels. The annotations are provided in polygonal format, enabling both segmentation tasks and conversion into bounding boxes. TACO objects are labeled into 60 classes, which are grouped into 28 supercategories as shown in Figure 3.1, including the *Unlabeled waste* category for objects that are difficult to recognize or heavily occluded.

The figure shows that some supercategories are over-represented (e.g., *cigarettes* and *Plastic bag & wrapper*), while others are under-represented, with 17 supercategories having fewer than 10 entries each. The authors of TACO propose a 10-class subdivision, retaining 9 of the original supercategories and merging the remaining ones into a single class called *Other*. However, this subdivision still does not achieve a good balance in the dataset

To mitigate this problem, the TACO authors also proposes a 1-class subdivision, named TACO-1, which is the most commonly used annotation approach, where only one class, *litter*, is considered. Regardless of the number of classes used, TACO remains a challenging dataset due to the presence of very small objects (e.g., cigarette butts, bottle caps, ropes, strings, etc.) and transparent objects (e.g., bottles, glass, etc.).

Figure 3.3 shows the histogram and cumulative distribution of the diagonal length of the bounding boxes relative to the longest side of the image. The plots indicate that the most frequent bins are the smallest ones; specifically, the most common bin corresponds to a bounding box with a longest side of approximately 12 pixels in an image scaled to have a longest side of 640 pixels, highlighting the challenging nature of the dataset.

TACO dataset also offers a background classification using the tags described in Figure 3.4. These tags are not mutually exclusive, since single photo may contain different backgrounds. While this classification is not strictly useful for the object detection task, it may be beneficial for splitting the dataset, for example, in a Federated Learning context.

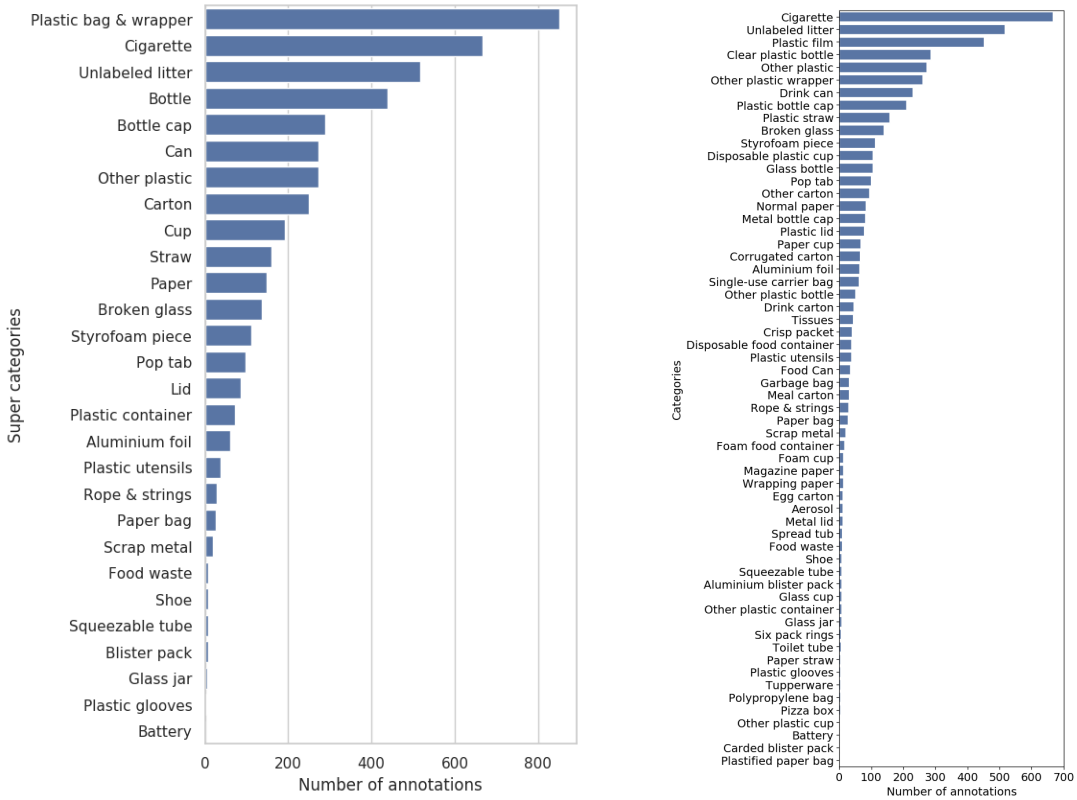

Figure 3.1: Number of annotations per super categories **(left)** and per categories **(right)** provided by TACO's current version. Images taken from [26]

Figure 3.2: Some examples of images from TACO dataset

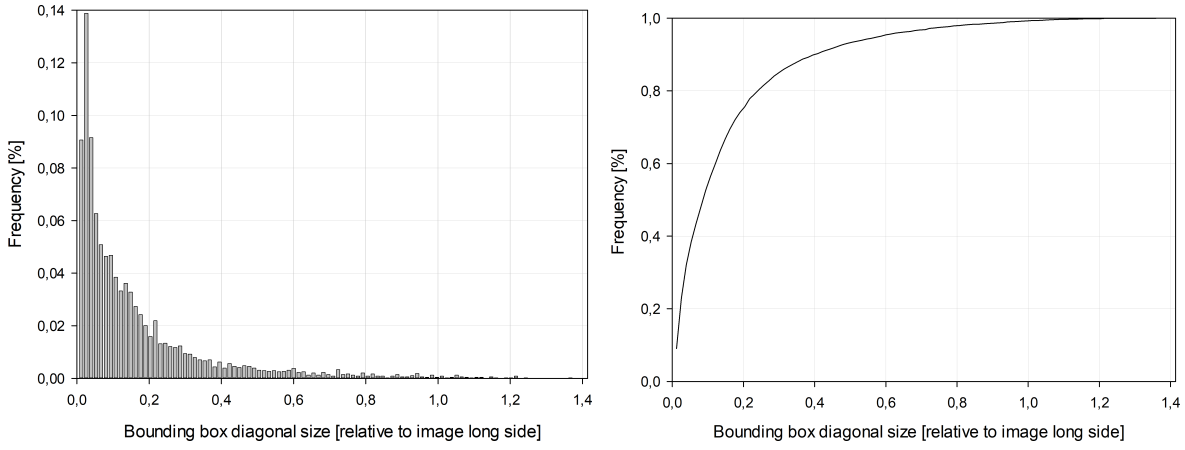

Figure 3.3: Histogram **(left)** and cumulative distribution **(right)** of the annotated bounding boxes in the TACO dataset, measured as the diagonal length of each bounding box relative to the longer side of the image.

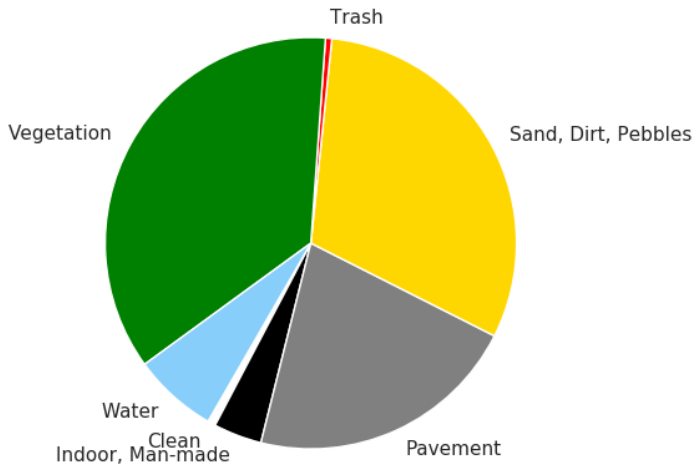

Figure 3.4: Proportion of images by background tag. Image taken from [26]

3.2 PlastOPol

The PlastOPol dataset, presented by Cordova et al. [7], is labeled with a single class, all of which fall under the *litter* category. This dataset comprises 2,418 images gathered by the Marine Debris Tracker, containing a total of 5,300 litter instances. Some examples are shown 3.5.

Each instance is enclosed within a rectangular bounding box defined by four values (x1, y1, width, and height), where (x1, y1) represents the coordinates of the upper-left corner of the bounding box. Compared to the TACO dataset [26], the PlastOPol dataset has larger bounding box areas, as shown in Table 3.2. Smaller bounding boxes mean

Dataset	**Images**	**Small**	**Medium**	**Large**
TACO	1500	384	1305	3095
PlastOPol	2418	33	445	4822

Table 3.1: Area of bounding boxes. **Small** $\rightarrow$ area ≤ 322, **Medium** $\rightarrow 322 <$ area ≤ 962, **Large** $\rightarrow$ area > 962. Table taken from [7]

fewer pixels, hence less information about the object to be detected: this means that PlastOPol could perform better than on TACO, having much larger objects to detect. Furthermore, the lighter models of YOLO v5 and YOLO v8 suffer particularly in detecting small objects, however much they may be present in the dataset.

The dataset presents images involving diverse types of environments as background, such as water, snow, sand, flint fields, streets, and more, as well as various types of litter, including plastic, glass, wood, metal, paper, cloth, and rubber. Moreover, PlastOPol contains images captured under different lighting, occlusion, and background conditions, resulting in very challenging detection scenarios. The authors of the PlastOPol dataset assert that these factors make it very comprehensive and ideal for training object detection models. Detecting litter composed of various materials, colors, and shapes against different natural backgrounds with varying lighting conditions increases the likelihood of misidentifying or failing to detect litter, especially in the presence of small litter instances.

Figure 3.5: Examples from PlastOPol dataset with different types of litter (a), types of environment (b) and different natural background (c). Image taken from [7]

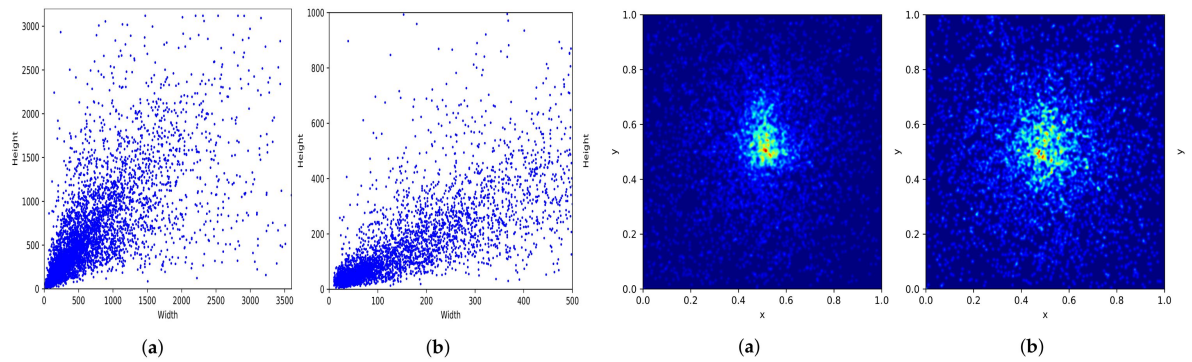

Figure 3.6: Bounding boxes by size **(left)** of PlastOPol **(a)** and TACO **(b)**. Bounding boxes by location **(right)** of PlastOPol **(a)** and TACO **(b)**. Images taken from [7]

Figure 3.7: Images from TACO **(above)** and PlastOPol **(below)** show waste in environments that are not really representative of real-world conditions. These *contrived* scenes oversimplify the context, potentially boosting model performance on the dataset, but not necessarily translating to improved results in real-world scenarios.

3.3 UniMiB Trash dataset

The decision to collect a dataset from scratch come 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.

3.3.1 Reasons to collect 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

As shown in Figures 3.7, 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

Figure 3.8: Images from TACO and PlastOPol **(above)** typically depict scenarios with only one or a few rejections. Some images from the UniMiB Trash dataset **(below)** present much more complex scenarios.

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, more so than PlastOPol, 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.

TACO has 4,784 annotations (about 3.19 per image), while PlastOPol has 5,300 annotations (about 2.2 per image). These datasets predominantly feature less complex scenarios, as shown in Figure 3.8. 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.

Single class or unrealistic classes

As previously introduced in Chapter 3, TACO includes 60 classes, but due to the dataset's size and characteristics, these classes are not fully exploitable. Certain features, such as distinguishing between different types of paper or plastic waste, are difficult to extract from images where the objects are smaller than 100 pixels. To support a larger number of classes, more images and a carefully defined class structure would be necessary. Consequently, the dataset is often reduced to a single class. In contrast, PlastOPol contains only one class (litter), unless the annotations are manually edited to define new classes. While a single-class task can already be addressed using existing datasets, the challenge lies in providing additional information to achieve the final goal. If this information cannot pertain to the type of waste, an alternative approach is to categorize waste by size, which is also significant and challenging.

3.3.2 Collecting pipeline

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.

Currently, most of the dataset is collected using three different smartphones by the author of this thesis. Another seven people, each with their own devices, contributed to the dataset, each in their own capacity and in smaller quantities. The majority of the photos were taken in the Milan hinterland, encompassing both urban and rural settings, from the countryside to rivers. Additionally, some images in the dataset were captured in Cambodia and Hong Kong.

To further diversify the images in the dataset, collaborations were sought with non-profit associations and individuals. The initial goal was to collect around 10,000 images; however, without external support, gathering even 2,000 images proved to be an immense effort. For example, in the city of Milan in Italy, it is relatively easy to photograph individual pieces of garbage that change constantly on a daily basis. In contrast, more complex scenarios are typically found in suburban or rural areas, where litter remains static and uncollected for long time.

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. Examples are shown in Figure 3.10.

Figure 3.9: Different conditions across various locations are one of the key features of the UniMiB Trash dataset. In the first row, the images show a picture taken in Cambodia **(left)** and one captured in the countryside of Brianza **(right)**. In the second row, the images depict an urban context in Milan **(left)**, along the Villoresi riverbanks **(center)**, and on the stairs of the Hong Kong MTR station **(right)**.

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

Hybrid Annotation Process using SAM

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) [19] 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. 4 class types are defined: *micro*, *small*, *medium*, and *large*. The classification is based on the 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. Some examples are shown in Figure 3.11.

The *small* class includes objects like common plastic, glass bottles and as 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., sheet metal, iron objects, mattresses, etc.)

The *micro* class includes very tiny objects, which are not considered essential for achieving the main goals. Cigarettes are the most common examples, along with small plastic items, candy wrappers, and others, as shown in Figure 3.12. Detecting these

Figure 3.10: Some examples of images of UniMiB Trash dataset with different and suboptimal conditions. Light blurring **(left)**, low light condition with flash camera **(central)**, blurring and low light condition with low-end smartphone **(right)**.

small 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.

Split Dataset & Exporting

These are two *automatic* tasks: first, you need to decide the split percentages for the training, validation, and test sets, and then export the dataset in YOLO format. Another approach involves initially dividing the dataset into a training set and a test set, and then further splitting the training set into K equal folds. This method ensures that the performance of the models remains independent of the specific split used.

3.3.3 Cluster problem

The addition of a new and fifth class in the dataset is being: the *cluster* class. As shown in Figure 3.8, annotating garbage that is stacked and indistinguishable from one another poses a challenge. The proposal of the *cluster* class aims to address dilemmas that have not been tackled in any other waste dataset thus far. An example of the proposed annotation is shown in Figure 3.13. In cases where objects within a waste accumulation are nearly indistinguishable, merging them into a single annotation may be the best solution. However, it is important to note that an Object Detector model might not perform well with this approach. Given that most existing datasets contain individual images, the model may tend to identify waste items separately rather than recognizing them as part of a larger accumulation. Furthermore, it would not be useful to obtain an accurate prediction of the bounding box of the cluster. Instead, the goal would be to determine the presence of the cluster and provide an estimate of its size: this

Figure 3.11: Examples of objects from different classes in the UniMiB Trash dataset are shown. In order: typical examples of the *small* class **(top)** segmented with a dark blue line, *medium* objects segmented with a light blue line **(middle)**, and *large* objects such as a mattress, armchair, and others segmented with a green line **(bottom)**.

Figure 3.12: Some examples of micro-class object taken from UniMiB Trash dataset. Sometimes it could be also difficult for human eye detect these objects.

is not measurable with the metrics used in Object Detectors, and may require the use of an ad hoc defined metric in the case of the segmentation task.

Nevertheless, this issue is critical to the success of the project. It is highly realistic that many reports may contain waste in this condition. The cluster problem remains open and under evaluation; it represents a challenging and novel problem to address, and any errors in this context could prove costly, particularly in terms of redoing annotations. Careful consideration and rigorous testing will be necessary to develop an effective approach to this dilemma without compromising the integrity of the dataset.

3.3.4 Dataset obtained

At present, the UniMiB Trash dataset comprises 1,232 annotated images, with 25 of those containing instances of clusters that are not yet suitable for model training. Currently, there are approximately 200 images awaiting annotation, indicating that the dataset is continuously evolving. The images in this dataset boast very high resolutions, similar to those found in the TACO[26] and PlastOPol[7] datasets, as illustrated in Figure 3.14. 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 Figure 3.15. 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

Figure 3.13: Image from UniMiB Trash dataset. Cluster annotation is the light blue one: it is impossible to distinguish and annotate every single waste or part of waste inside the cluster.

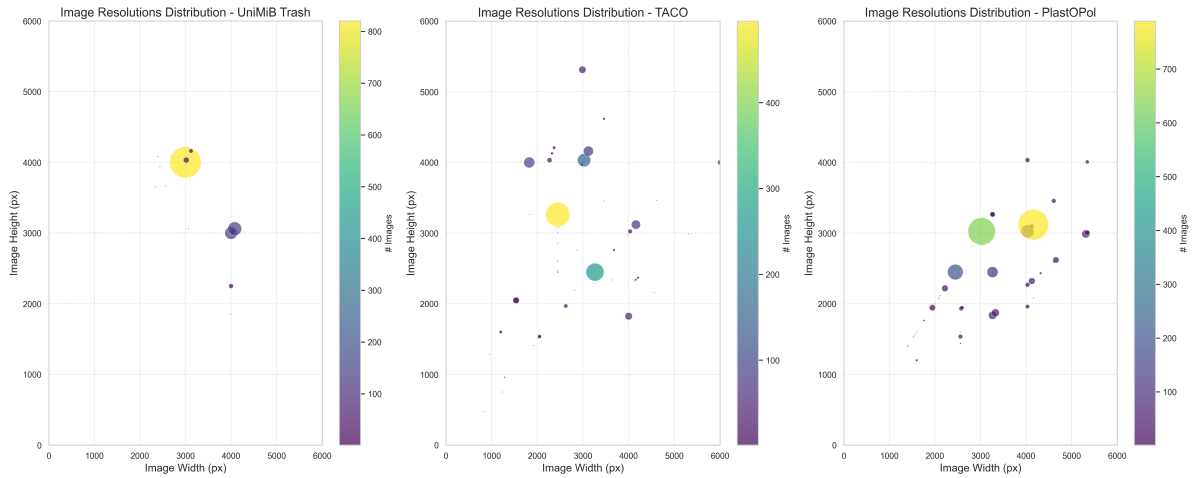

Figure 3.14: Distribution of resolutions of the three datasets used: UniMiB Trash **(left)**, TACO [26] **(middle)** and Plastopol[7] **(right)**. All three datasets have very high resolution images, despite the fact that the current models mostly deal with 640-pixel images. This makes these datasets possibly usable in the future, in case the models use images with higher inputs. It also gives the possibility of making crops to possibly synthetically augment the dataset.

Figure 3.15: Distribution of dataset instances for each class. The dataset is unbalanced, consistent with the analyzed reality in which larger waste items, such as bulky waste, are fortunately much less prevalent than small litter.

image. Similarly, a cigarette will always be categorized as "micro," regardless of whether it appears prominently or minimally within the frame. Figure 3.16 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 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.

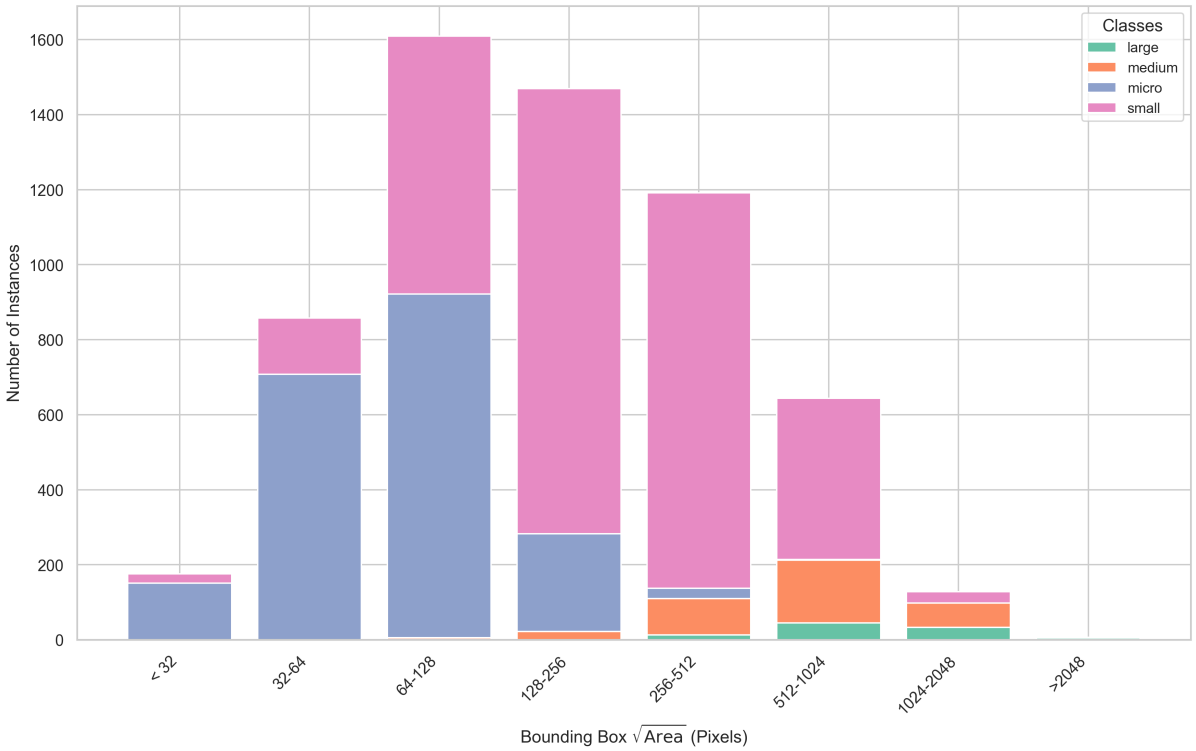

Figure 3.16: Distribution of the areas of the instances in the various classes in UniMiB Trash dataset. The consideration of object size is scale invariant, whereby an object is considered large or small according to its actual size.

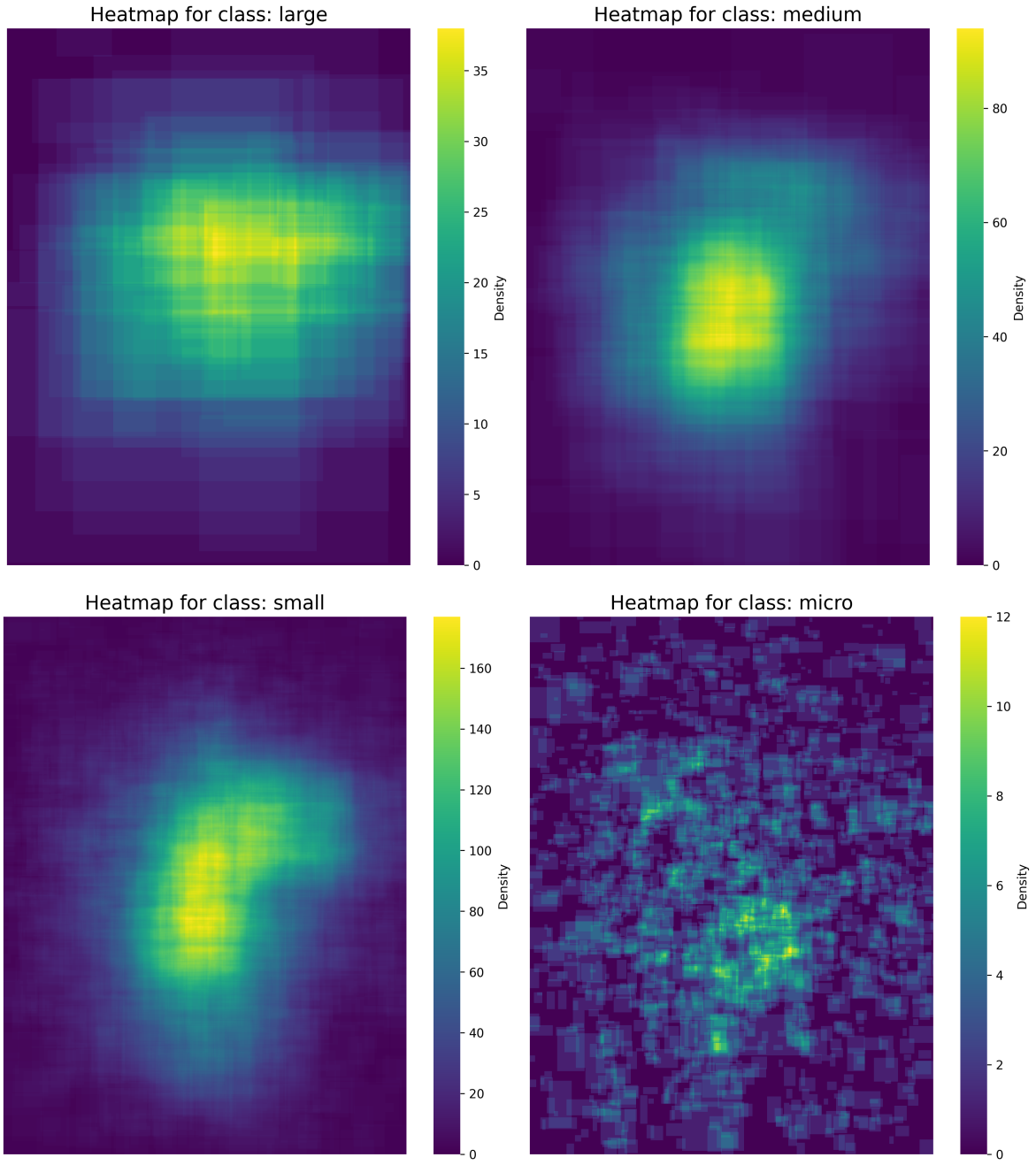

Figure 3.17: The heatmap above illustrates the spatial distributions of all objects for each class of the UniMiB Trash dataset. These visualizations offer insights into the most probable and rare locations of objects within the images. Among the classes, the micro class stands out as the most diverse and complex. It contains the smallest objects, which are often challenging for humans to detect, and the instances within this class are scattered throughout the images.

3.4 Wider Face

The need to find a dataset of faces arose because the final model must be able to censor any faces that may be unintentionally present in the waste images. To address this requirement, the WIDER FACE dataset is selected.

The WIDER FACE dataset [41] is currently one of the largest datasets for face detection, consisting of images selected from the publicly available WIDER dataset [40]. The authors chose 32,203 images and labeled 393,703 faces, retrieved using search engines like Google and Bing, resulting in a high degree of variability in scale, pose, and occlusion. The WIDER FACE dataset is organized into 60 event classes, with data randomly selected for training, validation, and testing sets in the proportions of 40%, 10%, and 50%, respectively.

The authors label the bounding boxes for all recognizable faces in the WIDER FACE dataset, ensuring that each bounding box tightly contains the forehead, chin, and cheeks. In cases where a face is occluded, the authors still provide a bounding box but include an estimation of the occlusion level. Similar to the PASCAL VOC [9] dataset, they assign an Ignore flag to faces that are difficult to recognize due to low resolution and small scale (10 pixels or fewer). After annotating the face bounding boxes, the authors further annotate additional attributes, including pose (typical or atypical) and occlusion level (partial or heavy). Each annotation is completed by one annotator and then cross-checked by two different individuals to ensure accuracy.

The images are grouped by event. To evaluate the influence of the event on face detection performance, the authors characterize each event using three key factors: scale, occlusion, and pose. For each factor, the detection rate is computed for the specific event class, and the results are ranked in ascending order. Based on this ranking, the events are divided into three categories: easy (41-60 classes), medium (21-40 classes), and hard (1-20 classes). This rank is calculated by three different parameters, which indicate the difficulty of predicting bounding boxes.

The first parameter is Scale. The images are classified based on the size of the faces in the images (measured by height in pixels) into three scales, as shown in 3.19: *small* between 10-50 pixels, *medium* between 50-300 pixels and *large*, over 300 pixels.

The second parameter is Occlusion. Occlusion is an important factor for evaluating face detection performance. The authors treat occlusion as an attribute and assign faces into three categories: no occlusion, partial occlusion, and heavy occlusion. Specifically, they ask annotator to measure the fraction of occlusion region for each face. A face is defined as "partially occluded" if 1%-30% of the total face area is occluded. A face with occluded area over 30% is labeled as "heavily occluded". Figure 3.20 shows some examples of partial/heavy occlusions. Detection rate decreases as occlusion level increases. The detection rates of faces with partial or heavy occlusions are below 50% with 8.000 proposals.

The third and final parameter is Pose. Pose is similar to occlusion. Two levels of pose deformation are defined: typical and atypical. Figure 3.20 illustrates examples of faces with both typical and atypical poses. Faces are annotated as atypical under two conditions: if either the roll or pitch degree exceeds 30 degrees, or if the yaw angle exceeds 90 degrees.

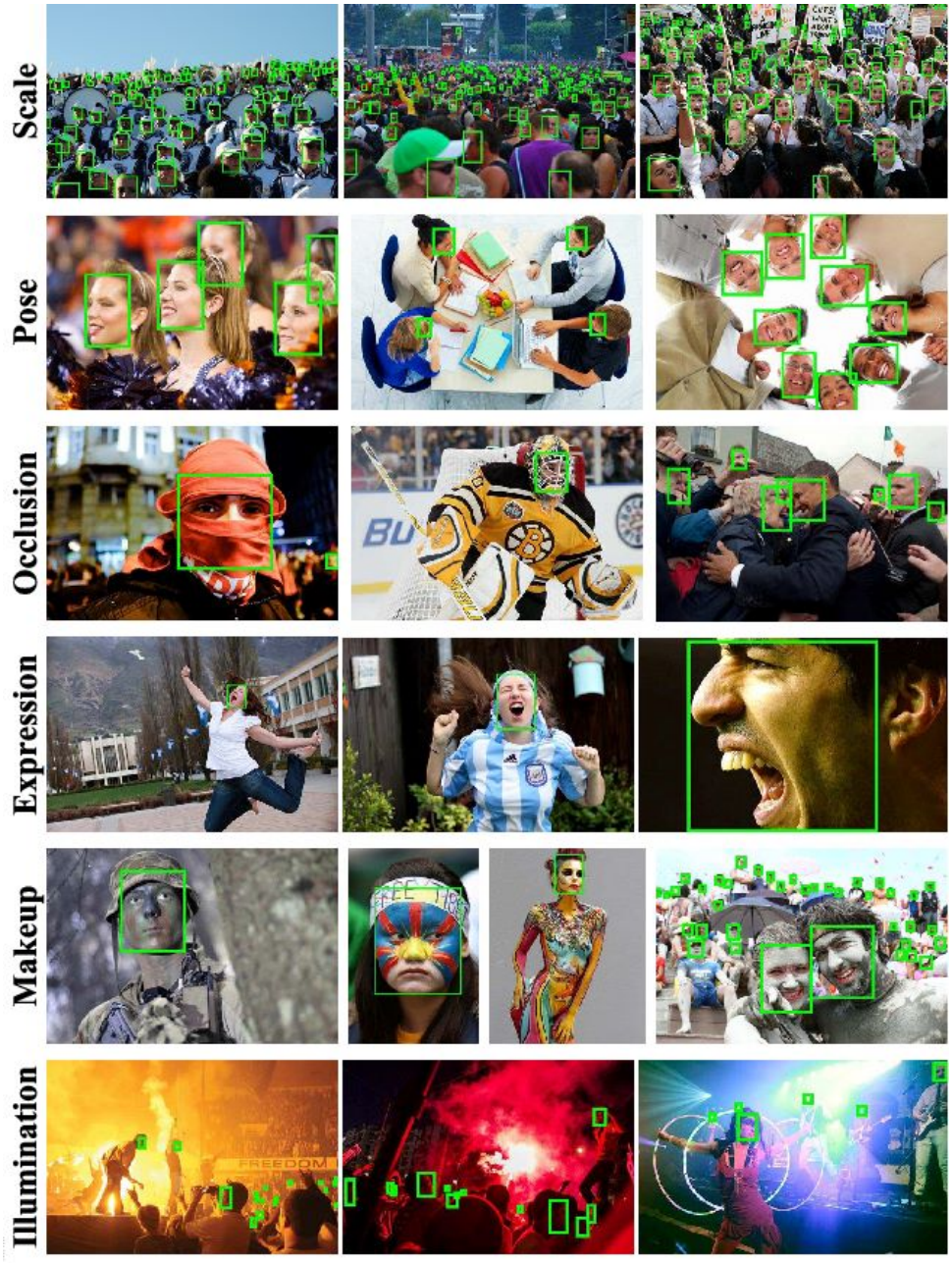

Figure 3.18: Some examples from WIDER FACE dataset. A high degree of variability can be observed in terms of scale, pose, occlusion, expression, appearance, and illumination. The annotated face bounding box is denoted by the green color. Image taken from [41]

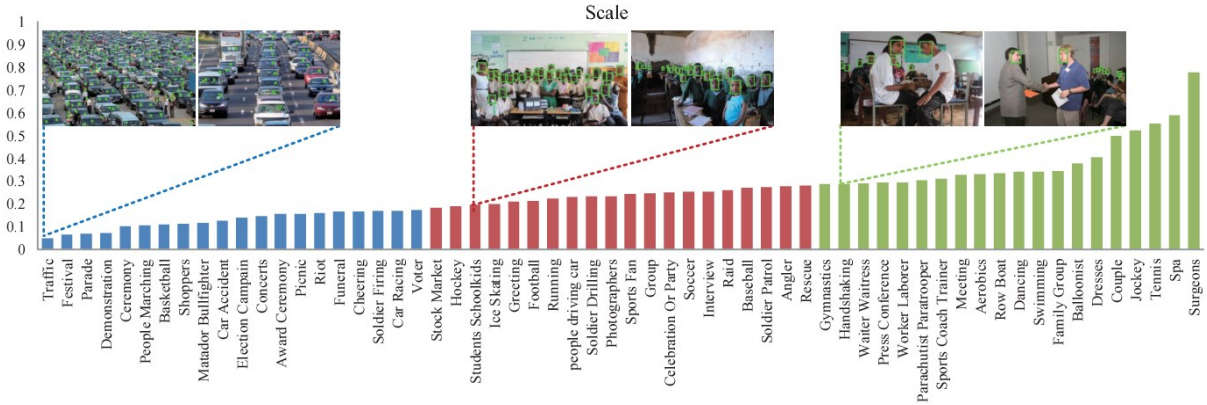

Figure 3.19: Histogram of detection rate for different event categories. Event categories are ranked in ascending order based on the detection rate when the number of proposals is fixed at 10,000. The top 1-20 event categories are denoted in blue, the 21-40 categories in red, and the 41-60 categories in green. Image taken from [41]

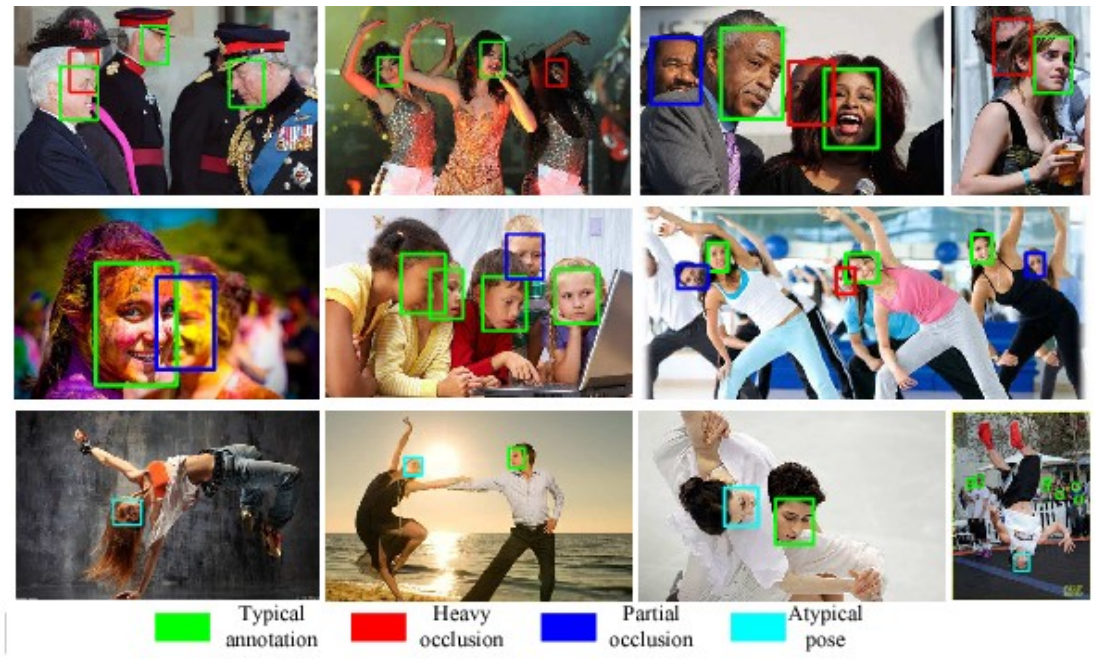

Figure 3.20: Some examples from WIDER FACE in different situation of occlusion. Image taken from [41]

Figure 3.21: Some example images taken from [5]

3.5 Mix License Plates

To censor another sensitive piece of information, the models must be capable of censoring car license plates. However, the available datasets are generally incomplete and not particularly well-suited for training deep learning models. The final dataset is a composition of the datasets presented as follows.

3.5.1 Car License Plate Detection

The Car License Plate Detection dataset [5] is designed for an object detection task and comprises 433 images containing a total of 471 labeled objects, all belonging to a single class called "license plate." However, the dataset lacks comprehensive documentation, and there are limited resources available for its use. The images in the Car License Plate dataset are of low resolution (less than 640 pixels) and present standard bounding box annotations in an XML structure. All images are labeled, but no training, validation, or test splits of the dataset are provided. As can be seen in Figure 3.21, the images are interesting and potentially suitable for training a deep learning model; however, the limited quantity of 433 images is insufficient for effective training. Despite this limitation, the dataset has been selected to be included in the combination of datasets that will be used to train the YOLO models.

3.5.2 UFPR-ALPR

The UFPR-ALPR dataset [21] is available exclusively for academic research and is provided free of charge to researchers affiliated with educational or research institutions for non-commercial use. This dataset comprises 4,500 images captured from within a vehicle navigating regular urban traffic. It is divided into 40% for training, 20% for validation and 40% for testing. The images were extracted from 150 videos, each lasting 1 second and recorded at a frame rate of 30 frames per second (FPS). They were captured using three different cameras and are available in Portable Network Graphics (PNG) format

Figure 3.22: Some examples taken from [21]. The camera framing is about the same.

Figure 3.23: An example of same shooting with 3 images 15 frames apart. The differences between this images are almost imperceptible, risking a strong overfitting phenomenon.

with a resolution of 1920 × 1080 pixels. Each image is accompanied by a text file containing annotations, which include details such as the camera used, the vehicle's position, and attributes like type (car or motorcycle), manufacturer, model, and year, as well as the identification and position of the license plate (LP) and its characters.

This dataset presents two significant problems. As previously mentioned, the training dataset is divided into 150 different shots, with each shot containing 30 images. These 30 images are extremely similar to one another, featuring the same setting and subjects, with only slight variations in framing, and sometimes appearing almost identical. This issue is illustrated in Figure 3.23. The second problem is that the camera is typically positioned inside the cockpit, resulting in similar framing across the images. These conditions create the perfect prerequisites for developing a model that conforms exclusively to the world depicted in the dataset, rather than the real world, which can be described with one word: overfitting.

To prevent this phenomenon, only one image per video has been selected, resulting in a total of 300 images extracted from the initial 4,500. While this number is not substantial, it is an improvement over nothing.

Figure 3.24: Example images taken from License Plates Dataset [32]. The type of images is similar approximately to the [5]

3.5.3 License Plates Dataset

The last dataset presented is the License Plates Dataset [32], which is a subset of the Open Images Dataset [20] focused on object detection for various vehicles (e.g., cars, vans, etc.) and their respective license plates, shared by Roboflow. The annotations also include examples of both vehicle and license plate. This dataset is divided into training, validation, and test sets with splits of 70%, 20%, and 10%, respectively.

As shown in Figure 3.24 with some examples, the images in this dataset exhibit slightly less heterogeneity compared to those in [5]. Similar to the previous cases, the total of 350 images is insufficient for effectively training a deep learning model.

3.5.4 Other license plates dataset

Now follows a description of some datasets considered but discarded, briefly describing their characteristics and reasons.

The Application-Oriented License Plate Recognition (AOLP) [17] contains 2,049 images of Taiwan license plates in various locations, time, traffic, and weather conditions. The whole dataset is categorized into three subsets: access control (AC) with 681 samples, traffic law enforcement (LE) with 757 samples, and road patrol (RP) with 611 samples. Each subset offers a good scope of samples to represent one of the three major applications. The background are really cluttered, with road sign and multiple plates in one image. RP refers to the cases that the camera is held on a patrolling vehicle, and the

images are taken with arbitrary viewpoints and distances. The dataset is accessible only by getting to the password. A written request was tried, but was never answered. The official website listed, at the time of writing this thesis, is not working.

Global License Plate Dataset [1] consists of over 5 million images, including diverse samples captured from 74 countries with meticulous annotations, including license plate characters, license plate segmentation masks, license plate corner vertices, as well as vehicle make, colour, and model. The authors also include annotated data on more classes, such as pedestrians, vehicles, roads, etc. The dataset comprises comprehensive annotations, including the precise location of the four corners of the license plates, the polygon points outlining the instance segmentation of the license plate, and the bounding box information. The dataset includes valuable information the annotation for four vertices provides the exact (x, y) coordinates, accurately representing the borders of the license plate for robust object segmentation. Siddharth Agrawal also quantify license plate blurriness, quantified through Variance of the Laplacian transform of the Image. This dataset might be the best existing dataset in the state-of-the-art for license plate recognition, but annotations are not available. The author was solicited several times, but the author never uploaded the ground truths needed to use the dataset.

MediaLab dataset is a research dataset of Greek license plates. It contains 706 images with alphanumeric characters, and does not provide an official train/validation split. Actually the dataset site is unreachable and the authors, however, do not provide the annotations.

SSIG-SegPlate [14] dataset is composed of 2.000 Brazilian license plates, roviding 101 on-track vehicles captured during the day. The video was recorded using a static camera in early 2015. The images of the dataset were acquired with a digital camera in Full HD resolution and they were release in PNG format. The average size of each file is 4.08 Megabytes (a total of 8.60 Gigabytes for the entire dataset). The Brazilian license plate has the size of 40cm $\times$ 13cm, which means that its aspect ratio is approximately 3.08. In the dataset, the license plates have sizes varying from 68 $\times$ 21 pixels to 221 $\times$ 77 pixels. This dataset is also impossible to download: the indicated download links no longer work.

RodoSol-ALPR [22] contains 20,000 images captured by static cameras located at pay tolls owned by the Rodovia do Sol (RodoSol) concessionaire, which operates 67.5 kilometers of a highway (ES-060) in the Brazilian state of Espírito Santo. There are images of different types of vehicles (e.g., cars, motorcycles, buses and trucks), captured during the day and night, from distinct lanes, on clear and rainy days, and the distance from the vehicle to the camera varies slightly. All images have a resolution of 1,280 $\times$ 720 pixels. The problem with this dataset is that it was collected with a static camera, so the framing is always the same: it does not at all simulate the desired scenario in this work. For this reason it was discarded.

Chapter 4

Methods

This section presents two innovative techniques aimed at enhancing the prediction capabilities of YOLO models and proposes a potential new paradigm with the only OVD available in the YOLO family. First, a method for tuning the Non-Maximum Suppression (NMS) algorithm, employed by all YOLO models (with the exception of the new YOLO v10, which is free of NMS), is introduced. This approach leads to a significant increase in performance on the TACO dataset and, more generally, across all other datasets. The second method involves an alternative approach to making predictions, which is based on the fundamental concept of the chain of thoughts: instead of generating an immediate single response, the model is guided by intermediate steps that contribute to the final output. This method is already employed in large language models (LLMs), and although the application differs, the underlying logic may be quite similar.

4.1 Thresholds Tuning

Usually, the tuning of hyper-parameters focuses on the value and algorithm of Learning rate, Batch Size, Activation Function, Regularization, Augmentation techniques, etc. These hyper-parameters can also be optimised with YOLO. However, there may be more specific hyper-parameters that can have a more significant impact. In fact, Figure 4.1 shows that by changing a specific hyperparameter with YOLO models, the detection could improve.

This section discusses a new approach, or rather, a method that may have been overlooked by researchers conducting similar studies. This method has contributed to the publication of [33] and has enabled surpassing all previously obtained results regarding the TACO dataset, establishing this work as a new state-of-the-art.

This method involves tuning the Confidence (conf) parameter and, secondarily, tuning the Intersection Over Union (IoU) parameter. To discuss the tuning of these parameters, it is essential to understand the algorithm of Non-Maximum Suppression (NMS) [15]. The NMS algorithm can be summarized as follows:

1. Initial Detection: YOLO outputs multiple bounding boxes for each object, each with a confidence score and class probabilities.

2. **Filter by Confidence Threshold**: Predicted bounding boxes are filtered based on the Confidence threshold parameter.

Figure 4.1: The topic of discussion in this section: The result obtained without having fully exploited all the steps of Non-Maximum Suppression **(left)** and the result obtained by tuning and choosing the most appropriate thresholds which are fundamental in the bounding boxes selection process **(right)**.

3. Sort by Confidence: the remaining bounding boxes are sorted by their confidence scores in descending order.

4. Compute Intersection over Union (IoU): for each pair of boxes, the box with the highest confidence score is compared with the remaining boxes, calculating the Intersection over Union (IoU) to determine the extent of overlap between two bounding boxes. This metric will be explained in the next chapter but can be summarized as the area of the intersection divided by the area of the union of the two boxes.

5. Suppress Boxes: if the IoU of two boxes exceeds the set threshold, the box with the lower confidence score is suppressed, meaning it will be discarded from the list of detections.

Non-Maximum Suppression is a widely used algorithm to suppress bounding boxes; however, there is one particular setting in the Ultralytics libraries that could compromise performance: the confidence score is set at 0.001, which is too low. This means that the suppression of the boxes is entrusted only to IoU threshold. While this typically does not adversely affect Recall, the model's ability to identify waste with precision is significantly diminished because it assumes that if one bounding box is isolated from the others—even with a very low confidence score—it must be retained. This often leads to numerous false positives. Additionally, relying solely on IoU can still minimally affect Recall, even if only slightly. If the NMS algorithm was designed with two thresholds instead of one, it would be worthwhile to experiment with both and analyze the results.

Figure 4.2 shows a clear example of the result of the NMS algorithm with confidence at 0.001, as set in the Ultralytics libraries and still too low, and then with setting the confidence threshold to a value of 0.3.

The tuning of the Confidence and IoU parameters is conducted after the model training and is performed on the validation set by exploring a range of potential values. Regarding the tuning of the Confidence threshold, the optimization of its value is carried out on the validation set by investigating the optimal value within the following range:

$$\mathrm{C} = \{0.001, 0.1, 0.2, 0.25, 0.3, 0.4, 0.4, 0.5, 0.6\}$$

For most of the experiments conducted, the optimal range for the confidence threshold is between $[0.15, 0.25]$. Furthermore, it is essential to exercise caution when significantly increasing this value, as the validation set may not fully represent the test set or real-world scenarios.

The ideal outcome is to achieve the highest mean Average Precision at IoU 50 (mAP50) without negatively impacting the initial Recall or, ideally, even slightly improving it. To make a good trade-off between these metrics, it is necessary to experiment with tuning the confidence threshold in conjunction with the IoU parameter. The tuning for the IoU parameter is performed within this range:

$$\mathrm{IoU} = \{0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.3, 0.2, 0.1\}$$

Typically, the best results are achieved with values in the range $[0.5, 0.2]$.

It is important to note that, unlike the confidence threshold, a lower IoU parameter is more stringent, because bounding boxes are removed if their IoU is greater than the specified threshold. For example, setting a threshold of 0.001 may lead to a significant loss of recall, while a threshold of 0.9 would allow nearly all bounding boxes to remain.

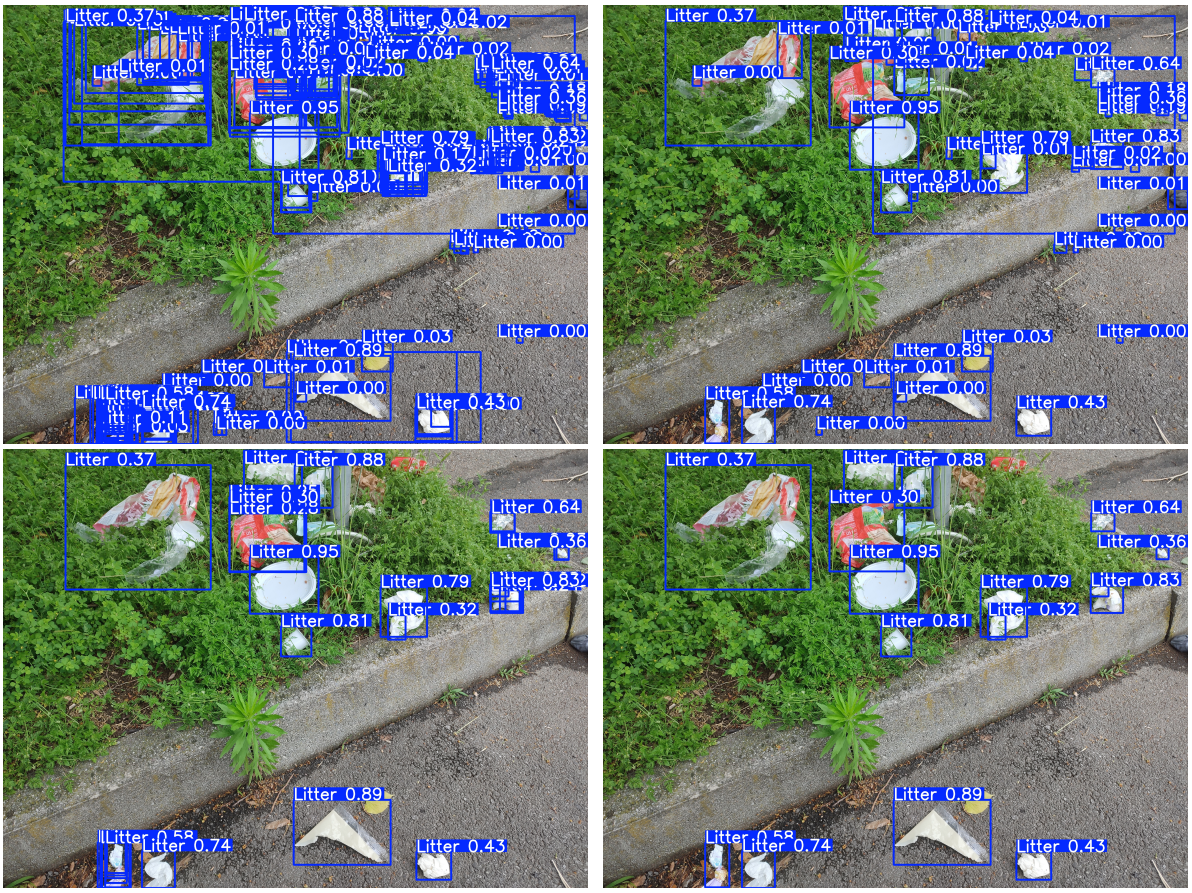

Figure 4.2: **(Above)** The first image displays the bounding boxes predicted by YOLO v8 for the *litter* class, removing predictions that have a confidence score lower than 0.001. Then, the second image shows the remaining predictions, where bounding boxes with an IoU greater than 0.3 have been removed. While some bounding boxes are eliminated in the bottom left of the picture, where there are a significant overlap between bounding boxes, incorrect predictions with very low confidence remain, as they are isolated and do not overlap with other predictions.
(Below) In sequence, the image with the bounding boxes predicted by YOLO v8 on *litter* class, now removing the bounding boxes with a Confidence score less than 0.3; then, image with remained bounding boxes after the IoU evaluation process, with IoU = 0.3. The final predictions are much better than the previous, taking advantage of the combination of the values of both thresholds.

Moreover, if n corresponds to the number of elements C (elements of Confidence range), tuning the IoU parameter increases the computational cost from linear time $O(n)$ to $O(n \cdot m)$, where m represents the number of elements in the IoU range. Given that the sizes of both ranges are similar ($n \approx m$), the time complexity can be approximated to $O(n^2)$.

4.2 Two-Stage Detection With YOLO World

As described in Chapter 2, YOLO World [6] belongs to the open-vocabulary object detection (OVD) family, which includes models like Grounding DINO and F-VLM. These models aim to detect objects beyond predefined categories, showcasing the potential of vision-language modeling in object detection tasks. YOLO World addresses the limitations of traditional object detection methods by enabling detection beyond fixed categories, offering adaptability to new tasks, reducing computational overhead, and simplifying deployment on edge devices. However, leveraging this potential is challenging. This section discusses the issues encountered when using YOLO World, particularly in identifying small objects with the standard zero-shot prediction approach. To address these challenges, a novel methodology is proposed to improve the detection of license plates and faces.

4.2.1 The delicacy of YOLO World

The main challenge with YOLO World lies in its sensitivity to the input prompt, which involves simply defining the name of the individual classes rather than providing an extended description. This approach is powerful yet difficult to set: using one name over a synonym can lead to performance variations of up to 10 percentage points in terms of mAP50, such as preferring a term in US English versus UK English. This emphasizes the importance of prompt selection and the potential impact on detection outcomes, making the configuration of YOLO World both a significant strength and a key area for optimization. But which rules to follow? It is not clear.

A simple example can be observed in Figure 4.3: where changing the prompt from *license plate* to *car license plate* yields different detection results. This illustrates the lack of consistency in how the model responds to various prompts: while some practical tips and community recommendations can be found online to improve predictions, there is still a gap in scientific publications and comprehensive research that could explain and better understand the behavior of YOLO World. This unpredictability highlights the need for further studies to establish guidelines and best practices for prompt configuration.

However, two images are not sufficient to demonstrate this. Therefore, Table 4.1 presents experiments conducted on the Mix License Plate dataset, described in chapter 3. As can be seen, YOLO World (in this case, the X version) gets little to very different performance depending on the classes set for detection. Consider that the "car" class is part of the COCO dataset; compared to other possible classes (e.g., litter), YOLO World performs very well, showing a very good values that will be discussed in the next chapter. Performance changes greatly depending on the input, and a few tips have been discovered that will be discussed in the next section.

License plate Car license plate

Figure 4.3: Examples from the UFPR-ALPR dataset [21]. They illustrate the effect of prompt variations on prediction outcomes using YOLO World X. When setting the class name to "license plate" (**left**), the model sometimes fails to detect objects that are correctly predicted when using the slightly modified class name "car license plate" (**right**). In general, predictions with "car license plate" (**below**) show higher confidence compared to those made with just "license plate" (**above**).

Table 4.1: Some examples tested with the license plate detection task using different prompts on the Mix License Plate dataset. The Null class is critical for successful detection, and the class *Car license plate* generally performs better than just *License plate*.

Method	mAP50	mAP50-95	Classes	Null class
Yolo World X zero shot	3.4	2.0	License plate	no
YOLO World X zero shot	47.4	21.2	License plate	yes
YOLO World X zero shot	4.2	2.5	Car license plate	no
YOLO World X zero shot	57.0	25.6	Car license plate	yes
YOLO World X zero shot	3.3	1.9	License plate + Litter and Face	no
YOLO World X zero shot	54.4	24.9	License plate + Litter and Face	yes
YOLO World X zero shot	3.8	2.2	Car license plate + Litter and Face	no
YOLO World X zero shot	60.1	26.8	Car license plate + Litter and Face	yes

Table 4.2: YOLO World performance on License Plates dataset with *Car License Plate* class, where increasing Confidence value does not compromise performance, as is the case in most of the YOLO models discussed in this paper. Rather, intervention on the IoU parameter is more effective; in this instance, a unique behavior is observed: restricting the parameter (lowering its value) results in a higher Recall than expected.

Method	mAP50	mAP50-95	Recall	Precision	Conf	IoU
YOLO World X zero shot	57.0	25.6	64.9	50.4	def	def
YOLO World X zero shot	57.7	28.2	64.9	50.4	0.2	def
YOLO World X zero shot	59.8	27.0	71.4	50.6	def	0.1

4.2.2 Tips to improve YOLO World performance

Low Confidence.

Unfortunately, the tuning described in the previous section regarding the NMS algorithm cannot be exploited.

YOLO World, unlike traditional Object Detectors, generally predicts bounding boxes with very low confidence values. Therefore, Confidence tuning should not be done, as it can kill Recall. However, this argument does not always apply; for some classes, it can predict with higher confidence, while for others, it predicts with very low confidence. The table provides an example where Confidence tuning could be beneficial. Two examples are shown: the first in Table 4.2, where tuning confidence does not compromise performance in terms of Recall, and the second in Table 4.3 where the increase in recall totally undermines the model's ability to detect objects of the desired class. In general, tuning the IoU might be better than tuning the Confidence in most scenarios with YOLO World, although it is much more complex than tuning with the more classic YOLO v5 and v8. Generally, since YOLO World is used for multi-class problems, as in this work, it is better to keep the Confidence parameter low and instead focus on adjusting the IoU to increase the model's accuracy, if necessary."

Table 4.3: YOLO World performance on TACO dataset with 1 class (Litter). Increasing of Confidence parameter compromises the model's ability to detect objects (Recall): high Precision is useless if the detected litter are almost zero; it is better try to tuning the IoU parameter to improve the model's Precision.

Method	mAP50	mAP50-95	Recall	Precision	Conf	IoU
YOLO World X zero shot	46.9	38.8	34.6	56.9	def	def
YOLO World X zero shot	50.9	46.2	**9.1**	93.2	0.01	def
YOLO World X zero shot	33.4	33.4	**0.2**	66.7	0.1	def
YOLO World X zero shot	50.1	41.1	33.1	65.5	def	0.1

Null class

A *null class* is a class in which you are not interested in performing detection, but asking a model to detect it is important because it often, though not always, improves the performance of the concerned class. However, even this is only moderately true: examples and counterexamples are shown in Table 4.4 and Table 4.5. In the Mix License Plate dataset, it is shown how the addition of the null class clearly improves performance; without it, the model cannot be considered able to detect the License Plate class.

Table 4.4: YOLO World performance on the TACO dataset with the *Litter* class, both with and without the *Null* class. Here, the performance of YOLO World must be observed carefully: without the *Null* class, it is more precise but achieves 6% less in terms of recall. As noted, without the null class, the model predicts with very low confidence, since Recall is drastically reduced by the increase of the confidence parameter.

Method	mAP50	mAP50-95	Recall	Precision	null class	Conf	IoU
YOLO World X zero shot	50.1	41.1	33.1	65.5	no	def	def
YOLO World X zero shot	36.1	22.7	39.7	50.5	yes	def	def
YOLO World X zero shot	33.4	33.4	**0.2**	66.7	no	0.1	def
YOLO World X zero shot	42.7	30.1	41.4	50.6	yes	0.1	def

Table 4.5: YOLO World performance on the license plates dataset with the *Car License Plate* class, both with and without the null class. The performance without the *Null* class is almost zero; thus, this class is needed. Furthermore, as seen in the previous table, with the null class, the confidence of the predicted bounding boxes increases, as raising the confidence threshold does not lead to losses in recall.

Method	mAP50	mAP50-95	Recall	Precision	Null class	Conf	IoU
YOLO World X zero shot	4.2	2.5	15.5	7.1	no	def	def
YOLO World X zero shot	57.0	25.6	64.9	50.4	yes	def	def
YOLO World X zero shot	57.7	28.2	64.9	50.4	yes	0.2	def

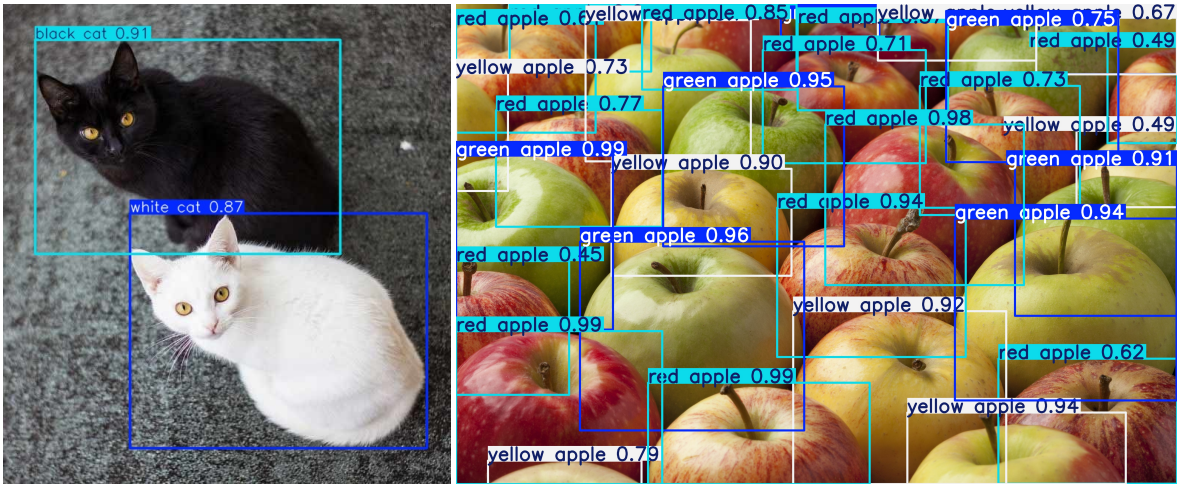

Figure 4.4: Some examples where YOLO World is able to distinguish classes based on color: **(Left)** A prediction of black and white cat classes and **(Right)** the prediction of three different types of apples based on their colors. This capability could be very interesting, but it is not useful for the tasks in this work.

Specify color

YOLO World has a strong sense of color; therefore, if classes can be distinguished by color, YOLO World should generally perform well. Two examples are shown in Figure 4.4. Unfortunately, in all the tasks addressed in this work, this capability is not useful for distinguishing classes from one another.

Describe the size of the objects

It seems that YOLO World has a good ability to distinguish objects based on their size. This is an interesting task that we tested on the "Litter" class using some images taken from the UniMiB Trash dataset. Two examples are shown in Figure 4.5: the results are not bad, but in many other images, YOLO World did not produce the expected results. It should be noted that this class is far from the pre-trained categories of YOLO World and that the scenarios are quite complex; however, this is what we will have to contend with in the UniMiB Trash dataset.

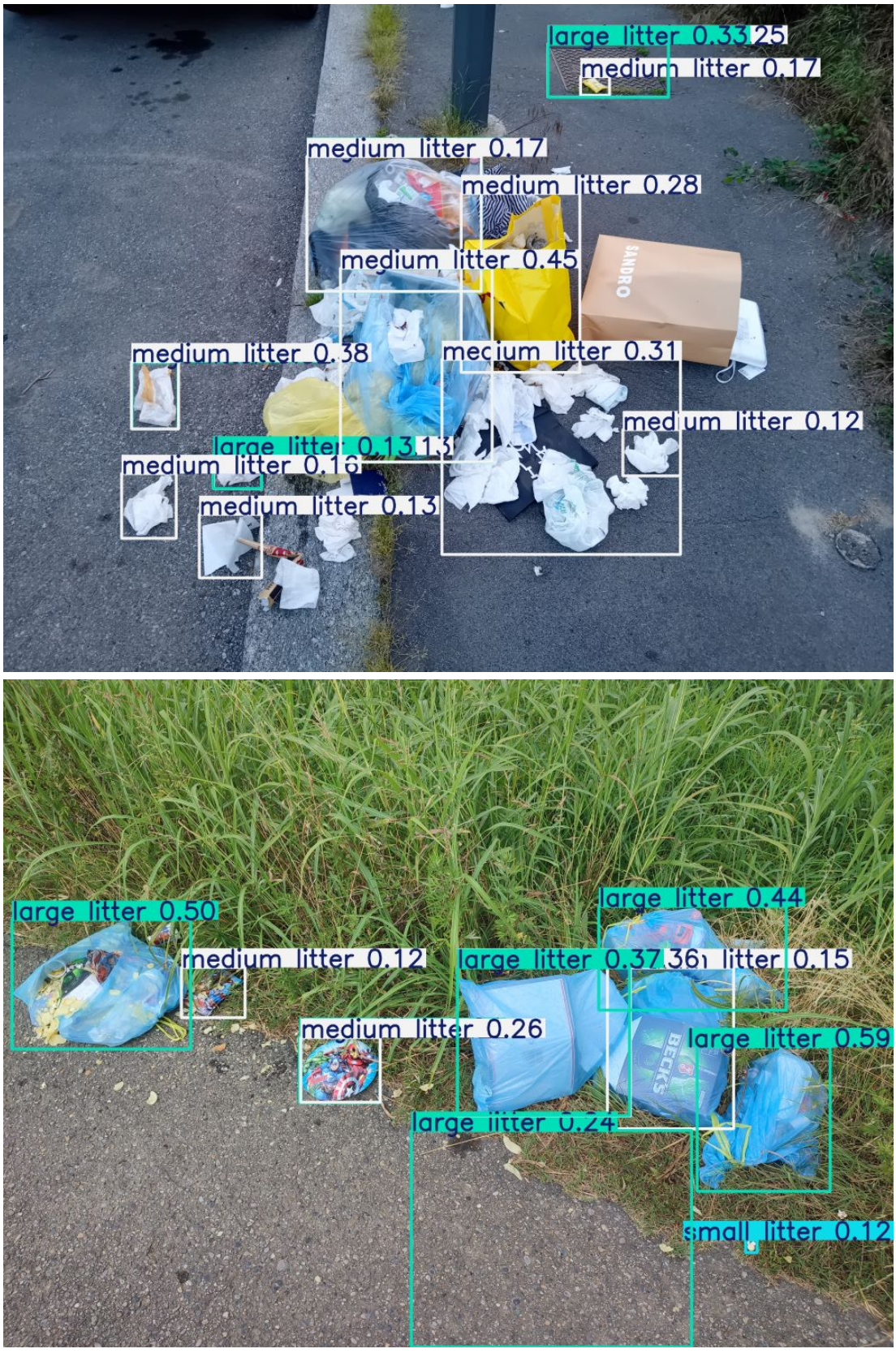

Figure 4.5: Some examples from UniMiB Trash dataset where YOLO World is able to distinguish classes based on the size. This ability might be useful in the multi-class task with the UniMiB dataset, where the distinction of classes is made on the basis of object size.

4.2.3 Two-Stage Method

It is not just a tip to try to improve the performance of YOLO World: it represents a slightly different way of thinking about prediction in object detection with YOLO World. This approach involves using the zero-shot ability with pre-trained weights to detect newly defined classes not directly, but through an intermediate step. The basic logic can be considered similar to Chain-of-Thought (CoT) [39], used in Large Language Models (LLMs), which introduces a series of intermediate reasoning steps, through the input prompt, that significantly enhance the ability of LLMs to perform complex reasoning.

With YOLO World, there is no prompting where to apply the the technique of Chain-of-Thoughts method, but the idea is to guide the model step by step, allowing it to compose the final answer incrementally.

The approach works as follows: instead of directly detecting desired classes which could have small objects, that can be challenging for YOLO models, the model first identifies a larger object that the target class may belong to. Then, it crops the area within the predicted bounding box and performs detection on the desired class within that crop. This assumes that the object we want to identify may be part of a larger object; however, this method might not always apply (e.g., detecting cigarettes, which are typically very small and not part of any larger object).

The ideal YOLO World model for this study should identify three classes: Litter, License Plate, and Face. Unfortunately, the Litter class cannot be considered at this time, but the License Plate and Face classes fit well into this method. Figure 4.6 shows the different steps involved in YOLO World's two-stage prediction process.

Like the Chain-of-Thought method used in LLMs, this two-stage detection approach with YOLO World is simple and clear. It helps guide the model's final output through intermediary steps without providing direct suggestions.

As Chain-of-Thought in LLMs, this two-stage detection method with YOLO World is very simple and clear, trying to compose the final output of the model without giving any suggestions, but through intermediate steps. In theory, this method appears promising, but the key question remains: does it perform well in real-world scenarios? Figure 4.7 shows the potential of this two-stage approach, effectively showcasing its performance. The comparison is made against two highly competitive models: YOLO World X, which predicts license plates directly at various resolutions, and YOLO v5n6u, which has been specifically trained on the Mix License Plate dataset. This result is quite remarkable: a model that has not been trained on the *License plate* class is able to outperform those specifically trained on a Mix License Plate dataset. Then, this strategy is applied to additional images taken from the UFPR-ALP and UniMiB Trash Datasets, as illustrated in Figure 4.8 It was decided to develop code to test this two-stage method on the test sets of the datasets related to faces and license plates, in order to obtain reliable and comparable data. There is visual evidence demonstrating that this method is effective on images considered realistic. It is important to evaluate this method by testing it and collecting the results. This includes comparing the two-stage approach and two methods: the YOLO models trained on the respective data sets and with YOLO World used with normal prediction on license plates and faces. Results from the Wider Face dataset (for the face class) and the Mix License Plate dataset (for the license plate class) will be highlighted. Full results, including detailed comparisons and metrics, can be found in Chapter 5. The extracts aim to provide an overview of the two-stage method's performance against other YOLO models. Before presenting the results, some empirically

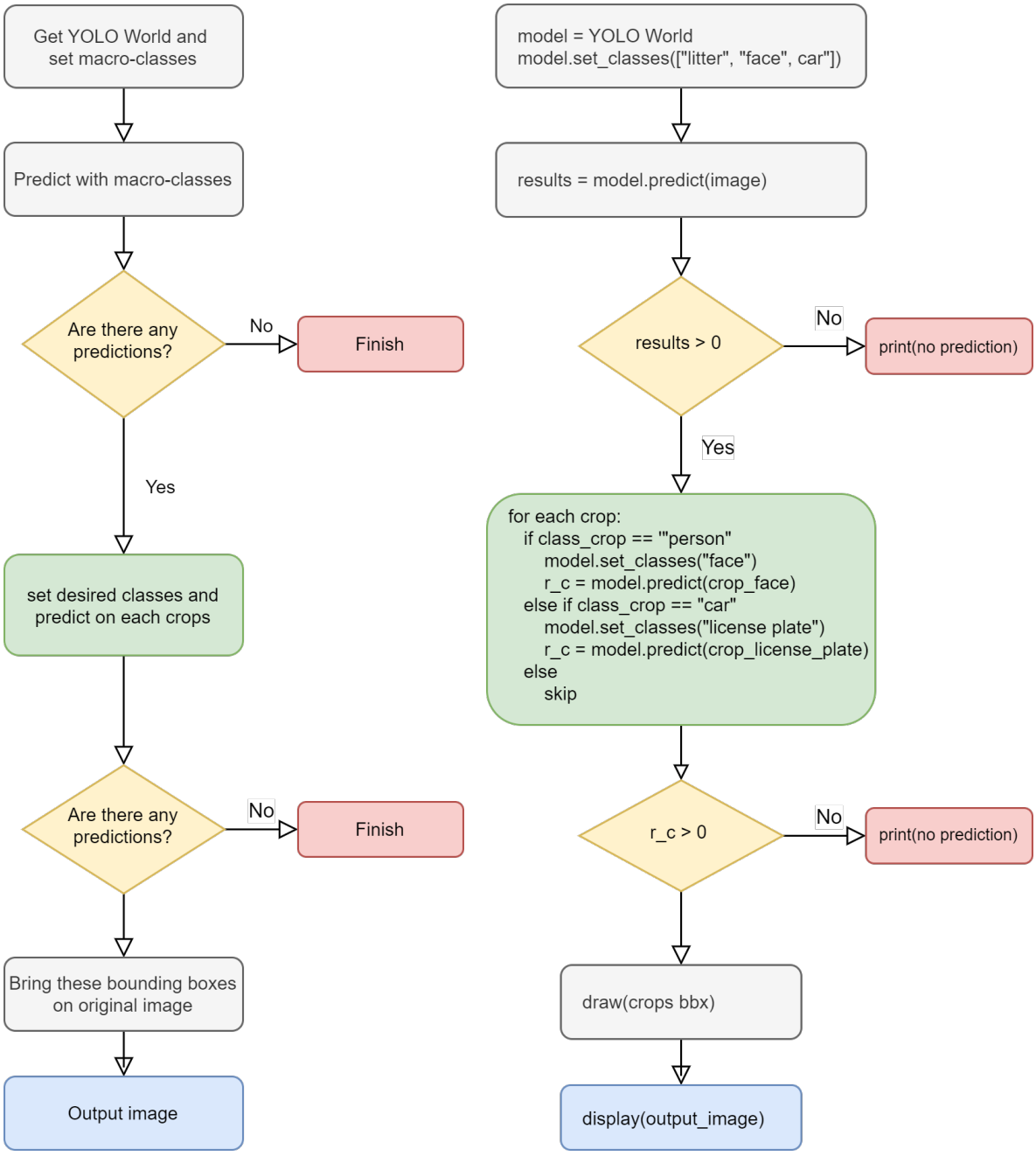

Figure 4.6: YOLO World two-stage prediction process workflow idea **(left)** and corresponding pseudocode **(right)**. Instead of offering suggestions to improve the prediction, the focus is on structuring the execution flow. This structured approach helps guide the model's process, potentially leading to more efficient and accurate results.

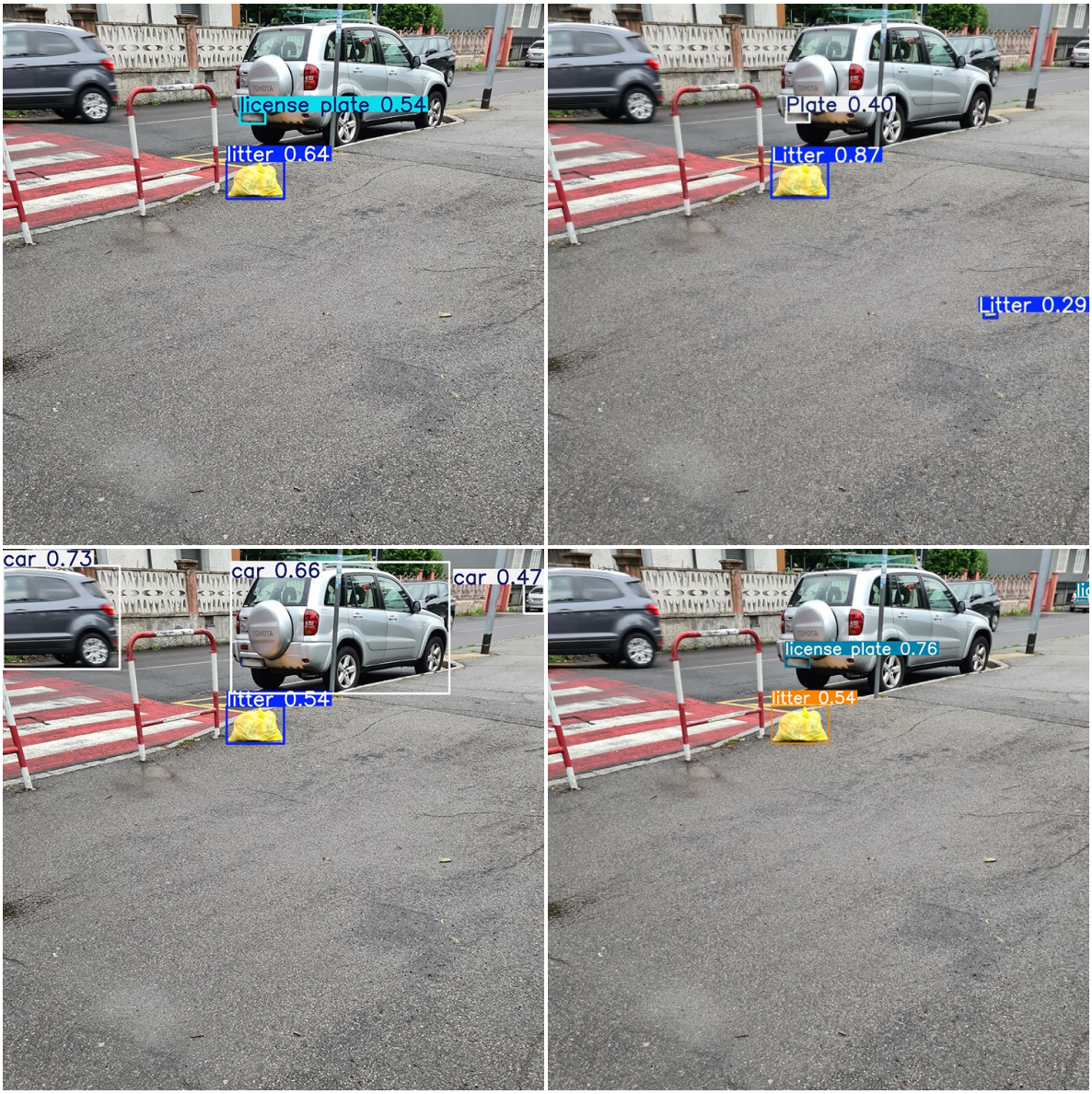

Figure 4.7: **(above)** The predictions made by YOLO World X **(left)** and YOLO v8s **(right)** for the *litter* and *car license plate* classes. The predictions are satisfactory with both, but it is important to note that the input image is of high resolution and has good properties (luminance, sharpness, etc.).
(below) The two-stage YOLO method: **(left)** image with predicted bounding boxes on *litter*, *person* and *car* macro-classes. **(right)** Image with final prediction on desired classes.Notably, the bounding box predictions exhibit higher confidence levels compared to those from other models, despite the lack of specific training on license plates. Furthermore, the model successfully detects a very small license plate located in the upper right corner achievement, that other models would likely not detect. This detection is facilitated by the initial identification of the car, which is a more manageable size, allowing for effective cropping and subsequent prediction of the license plate. This demonstrates the extraordinary capability of two-stage method.

YOLO v5n6u | YOLO World X | YOLO World two-stage

Figure 4.8: Comparison between two models with classical prediction (YOLO v5n6u and YOLO World X) and YOLO World X in two-stage mode. Excluding litter detection and concentrating only on license plates, the two-stage mode generally predicts the same license plates as the other models but with higher confidence. Additionally, it successfully detects even the smallest license plates, as shown in rows 3 and 4.

Table 4.6: Results of license plate detection on the test set of the Mix License Plate dataset, both with and without the various improvements presented. These methods are effective in reducing incorrect predictions, particularly when YOLO World mistakenly detect the entire car as a license plate in the cropped image.

Method	mAP50	mAP50-95	mAR50-95	Filter by size	Adaptive Crop resize
YOLO World X (two stage)	55.21	33.5	33.5	no	no
YOLO World X (two stage)	77.2	46.7	47.6	no	yes
YOLO World X (two stage)	78.0	47.3	47.7	yes	no
YOLO World X (two stage)	87.3	52.8	53.7	yes	yes

observed improvements will be discussed. These improvements are based on various tests carried out and confirmed by multiple non-authoritative online sources.

Filter Predictions by Size

During the predictions on the crops, for example when license plate or face classes is predicted on the corresponding crop, YOLO World sometimes incorrectly predicts very large bounding boxes, almost as if it were predicting the macro class itself. For example, after detecting a car and then making the license plate prediction, the model often returns a bounding box nearly as large as the entire crop (i.e., the car). These bounding boxes typically have high confidence values and do not overlap with other boxes, making them difficult to remove using standard methods. To address this issue, a simple filtering technique can be applied: if a bounding box has an area covering 85% or more of the crop (calculated using the width and height of the bounding box in COCO format), it should be removed. This ensures that only plausible-sized detections remain.

Do not resize the crop

Often, the crops are very small, with the longest side smaller than 100 pixels, and resizing by default resize to 640 pixels (keeping the original aspect ratio) can degrade the model's performance. To prevent this, the following resizing rule is applied: if the crop already has a longest side of 640 pixels or more, keep the original size, otherwise perform standard resize of the longest side to 640 pixels, maintaining the original aspect ratio. This approach aims to balance image quality and model performance by avoiding excessive scaling for very small crops. Table 4.6 illustrates the improvement regarding license plate detection is given. Average Recall is also reported to give an indicator regarding the model's ability to detect as many objects as possible (Recall metrics are not available with this method).

Once all these steps are implemented, the results shown in Tables 4.7 and 4.8 speak for themselves: the two-stage method clearly outperforms the classical prediction approach, and for license plates, it even surpasses a YOLO v8 model trained on the dataset's training set. Full results are available in Chapter 5, where all models trained and evaluated on the respective datasets are presented.

It is important to note a technical issue in comparing these results: the `val()` function from Ultralytics, used in normal conditions except the two-stage method, includes an additional preprocessing step that appears to add padding to all images to make them uniform in size. This could affect model performance, often favoring the `val()` method over the `predict()` method.

Table 4.7: License plate detection results obtained on the test set of the corresponding dataset. It is important to note that the two-stage methods use the `predict()` function from the Ultralytics library, while all other methods use the `val()` function. Many users have reported issues with the instability of comparisons between the two methods, which should ideally behave the same way; this discrepancy is due to a preprocessing step applied to the images, which adds padding during the resize operation. The performance of YOLO World Small with two-stage method is impressive: it outperforms YOLO v8s of this work, and its results are very similar to those of YOLO v8n, despite the fact that YOLO World Small in the two-stage approach was never trained on a license plate dataset. The performance of YOLO World Small and X with normal prediction is decent, but the gap (over 25% of mAP50 and mAP50-95) is not comparable with the other models.

	Result		Tuned Value	
Method	mAP50	mAP50-95	Conf	IoU
YOLO-v8n (this work)	90.0	57.2	0.2	0.5
YOLO-v8s (this work)	89.2	53.4	0.1	0.3
YOLO World S zero shot (1 class)	47.9	19.5	def	0.01
YOLO World S zero shot (4 classes)	43.2	17.0	def	def
YOLO World X zero shot (1 class)	57.0	25.6	def	0.01
YOLO World X zero shot (4 classes)	61.7	27.1	def	0.01
YOLO World S zero shot (two stage)	**90.3**	**56.0**	def	def
YOLO World X zero shot (two stage)	87.3	52.8	def	def

The two-stage method developed uses the official PyProtocols library with COCOeval, taking the predicted bounding boxes, comparing them with ground truths and performing the calculations of the metrics. Although there is no guarantee that the results are comparable to the exact thousandth, the two-stage method has been validated multiple times, supporting the claim of its superiority. Undoubtedly, the computational time differs: the two-stage method requires an initial prediction followed by a separate prediction for each crop, though typically using low-resolution images, which makes the process faster. However, it is important to note that this approach does not require any additional training to detect new classes, and YOLO World provides a light version, YOLO World small, which is able to run on smartphones. This means that, instead of relying on a large, complex model trained to provide an immediate solution to a challenging problem, we can use a lighter model without training, giving it more time to "reason", i.e. more intermediate steps to perform the prediction. Although it may take more time than the larger model, the process is guided by multiple steps, leading to a solution for a problem that is inherently complex.

This reasoning aligns with a recent publication by Google DeepMind researchers, Charlie Snell et al.[34], on LLMs models: the researchers claim that instead of using a larger model to generate a single response, it is more effective to use a smaller model to produce multiple responses, iteratively refining the final output. The approach proposed here with YOLO World is similar: might it not be better to use YOLO World Small in a two-stage mode, involving an initial prediction followed by intermediate predictions to refine the final output, rather than relying on a YOLO World X model with a single prediction?

Table 4.8: Face detection results on each Wider Face validation set. The performance of YOLO World with the two-stage method is lower than YOLO v8s. However, it is significantly better than the performance of the standard prediction approach, with an improvement of approximately 20% in terms of mAP50.

Method	Set	Default thresholds		Tuned Value	
		mAP50	mAP50-95	Conf	Iou
Faceness [41]	Easy	71.3			
YOLO-v8s (this work)	Easy	83.6	54.6	0.2	0.5
YOLO World S zero shot (1 class)	Easy	35.2	14.6	def	0.01
YOLO World S zero shot (two stage)	Easy	61.8	31.7	def	def
YOLO World X zero shot (1 class)	Easy	48.3	24.5	def	0.01
YOLO World X zero shot (two stage)	Easy	**66.0**	**39.9**	def	def
Multiscale Cascade CNN [41]	Medium	63.6			
YOLO-v8s (this work)	Medium	74.6	47.0	0.2	0.5
YOLO World S zero shot (1 class)	Medium	30.3	11.5	def	0.01
YOLO World S zero shot (two stage)	Medium	52.8	26.8	def	def
YOLO World X zero shot (1 class)	Medium	42.9	20.7	def	0.01
YOLO World X zero shot (two stage)	Medium	**63.8**	**37.2**	def	def
Two-Stage CNN [41]	Hard	58.9			
YOLO-v8s (this work)	*Hard*	*78.0*	*47.2*	*0.2*	*0.5*
YOLO World S zero shot (1 class)	Hard	31.8	12.6	def	0.01
YOLO World S zero shot (two stage)	Hard	**54.0**	30.0	def	def
YOLO World X zero shot (1 class)	Hard	44.6	22.0	def	0.01
YOLO World X zero shot (two stage)	Hard	50.4	**31.7**	def	def

Chapter 5

Experimental Results

5.1 Experimental Setup

This research leverages the TACO [26], PlastOPol [7], UniMiB Trash, Wider Face [41], Car License Plates [5], UFPR-ALPR [21] and License Plates [32] datasets, as delineated in Chapter 3. In the absence of local hardware availability, initial approaches to all problems were made on Google Colab with the free plan, which provides an NVIDIA Tesla T4 GPU with 15GB of vRAM, CPU Intel Xeon 2.20 GHz and 12 GB of RAM. The maximum duration of a GPU session is only 3 continuous hours: sometimes it is impossible to finish training of a model. When an approach is promising, it has moved from Google Colab to a local machine on which long model training can be done. 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.

5.2 Metrics

The different methods are compared in terms of mAP50, mAP50-95 (also known as mAP), Precision, and Recall. Before explaining these evaluation metrics, it is important to understand the concepts of labels, model outputs, and Intersection Over Union (IoU). Labels represent the ground truth annotations for the dataset. Each label includes two components. The first is bounding box, which is the location of the object in the image represented by coordinates usually in COCO format `(x1, y1, width, height)`. The second one is the class, the category of the object (e.g., litter, license plate, etc.).

When performing inference on an image, the model output consists of three elements: the predicted bounding boxes (also in COCO format), the class, and the confidence score for each bounding box. The confidence score, ranging from 0 to 1, indicates how confident the model is about its prediction.

The Intersection over Union (IoU) is a metric that measures how well the predicted bounding box matches the ground truth. IoU is calculated using the formula shown in Figure 5.1.

It should be noted that the ground truth in all tasks does not need to be perfectly precise, so high IoU values, without necessarily reaching the maximum value, can be sufficient to determine whether a detection has occurred. Typically, if the IoU is greater than a specified threshold (commonly set at 0.5), the prediction is considered a True Positive (TP); otherwise, it is considered a False Positive (FP). To better understand the

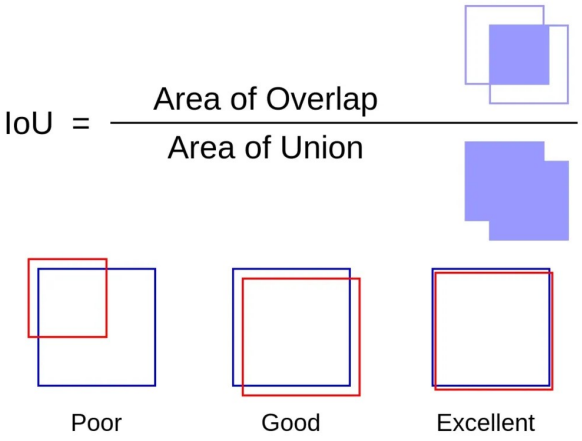

Figure 5.1: IoU formula. The *Area of Overlap* refers to the region where the predicted bounding box and the ground truth bounding box intersect, while the *Area of Union* is the total area covered by both boxes combined. This metric is useful for evaluating the overlap between the predicted bounding box and the ground truth.

		Actual value	
		p	**n**
Predicted value	**p′**	True Positive	False Positive
	n′	False Negative	True Negative

Table 5.1: Confusion Matrix with predicted value on the left and the Actual Value (ground truth) at the top.

distinction between TP and FP, a confusion matrix is shown in 5.1.

A True Positive (TP) occurs when the model correctly predicts a positive outcome, while a True Negative (TN) occurs when the model correctly predicted a negative outcome. A False Positive (FP) occurs when the model incorrectly predicts a positive outcome, and False Negative (FN) occurs when the model incorrectly predicts a negative outcome. As explained earlier, the correctness of a prediction is determined based on the Intersection over Union (IoU). A predicted bounding box could be considered a TP or FP depending on the IoU value. A False Negative (FN) refers to a ground truth bounding box that the model failed to detect but should have predicted.

Now the metrics used can be explained. Precision is the proportion of the all model's positive classifications that are actually positive. It is mathematically defined as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

If Precision is high, most of the model's predictions will be correct, although there may still be many False Negatives. The focus of this metric is on the model's ability to predict being as fussy as possible when making a prediction. Recall, also known as True

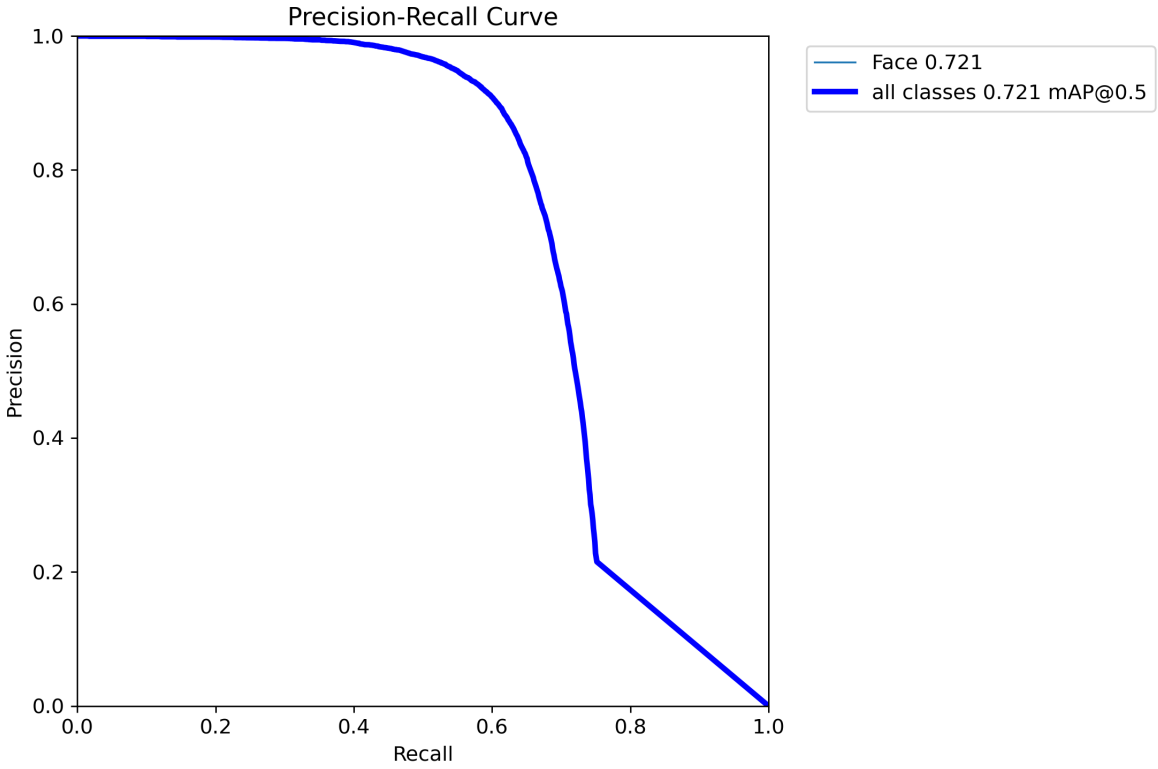

Figure 5.2: Precision-Recall curve. The curve starts near the top-left corner, indicating high Precision and low Recall. This means that at the beginning, the model is very precise but may not be capturing all positive instances. As you move to the right, recall increases, but precision may decrease. This is because the model starts to capture more positive instances, but it may also include more false positives.

Positive Rate, is the proportion of all actual positives that were correctly classified as positives. It is mathematically defined as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

A high Recall indicates that the model is able to detect most of the actual objects. This means the model may predict many objects, and a significant portion of those predictions could be incorrect while still maintaining a high Recall value; this occurs because False Positives are not taken into account. Achieving high values for both Recall and Precision is ideal, but often a trade-off must be made to balance the two metrics effectively.

This can be better understood through Precision-Recall (PR) curve, which is a graphical representation used to evaluate the performance of a classification model. In Figure 5.2 is shown an example, where Precision (y-axis) is plots against Recall (x-axis) at different threshold score. At lower confidence values, the model makes more predictions, which increases Recall but often decreases Precision due to the introduction of more False Positives (FP). Conversely, if the confidence threshold is increased, Precision improves (fewer FP), but Recall may decrease as fewer True Positives (TP) are detected.

Although the Precision-Recall curve is useful for evaluating the performance of a detector, comparing different models can be challenging when their curves are similar.

Average Precision measures the area under the Precision-Recall curve. It is a common metric for object detection models, evaluating how well the model maintains precision as recall increases. The AP score typically ranges from 0 to 1, where higher values indicate better performance. The simplified formula of Average Precision is as follows:

$$\text{AP} = \sum_{n=1}^{N} (R_n - R_{n-1}) \times P_n$$

where P_n is the precision at n-th threshold; R_n is the recall at n-th threshold; N is the number of thresholds.

The Average Precision can be computed at specific Intersection over Union thresholds, usually at IoU 0.5 (often denoted as $AP50$) and also across IoU thresholds from from 0.50 to 0.95 (denoted as $AP50-95$), in steps of 0.05. Specifically, it calculates the Average Precision at IoU thresholds $[.50, 0.55, 0.60, ..., 0.95]$ and then averages the results. Here are the formulas for both:

$$\text{AP}_{50} = \text{AP}(t = 0.5)$$

$$\text{AP}_{50-95} = \frac{1}{10} \sum_{t=0.5}^{0.95} \text{AP}(t)$$

Usually AP is computed for each class and then average across all classes to get mean Average Precision (mAP50).

$$\text{mAP} = \frac{1}{C} \sum_{c=1}^{C} \text{AP}_c$$

Where C is the number of classes. Based on the value of IoU, usually two type of Average Precision are distinct: mAP50 and mAP50-95. mAP50 refers to Average Precision at a single IoU threshold of 0.50, and is a good indicator of general object detection performance but may overlook errors in precise localization. mAP50-95 is a more comprehensive metric because it averages the precision across 10 different IoU thresholds. This metric is much stricter than mAP50 because it penalizes models that don't localize objects precisely at greater IoU thresholds (e.g., 0.75, 0.85, etc.). mAP50-95 provides a more nuanced view of a model's performance at different levels of IoU thresholds, allowing one to understand not only whether the model is detecting the object, but whether it is still very accurate (bounding boxes have a high IoU value) in detecting it.

Like AP, average recall (AR) is also a numerical metric that can be used to compare detector performance. In essence, AR is the recall averaged over all $IoU \in [0.5, 1.0]$ and can be computed as two times the area under the recall-IoU curve.

$$\text{AR} = \frac{1}{T} \sum_{t=1}^{T} R(t)$$

where T is the number of IoU thresholds and $R(t)$ is the Recall at threshold t. This metric is not used in the following experiments, but it was used in the two-stage YOLO World method in Chapter 4, in the absence of the Recall value provided by a third-part library to better understand the model performance.

Last but not least, the Efficiency is calculated in terms of model size in Megabytes (MB). A significant challenge in measuring efficiency based on processing time in frames per second (FPS) is its heavy dependence on the GPU used. Reliable results that can

be compared with state-of-the-art models can only be obtained locally or with dedicated professional cloud solutions. However, different researchers often present their results based on their own hardware configurations. Therefore, it was decided to report only the sizes of the different models. In YOLO models, size is also a general indicator of potential inference time. YOLO v5n6u and v5s6u models represent an exception, processes higher-resolution images (1280×1280 instead of 640×640).

During the models' training, three different type of Loss Function are used to monitor the performance of models on the validation set: Box loss, Class loss and Distributed Focal Loss.

Box LOss is the loss function used to measure the difference between the predicted bounding boxes and the ground truth. In this case, the Complete IoU (CIoU) metric is used, which not only measures the overlap between predicted and ground truth bounding boxes but also considers the difference in aspect ratio, center distance, and box size. Together, these loss functions help optimize the model for object detection by minimizing the difference between the predicted and ground truth class probabilities and bounding boxes.

Class Loss measures the error in predicting the class of the detected object. It is computed based on Binary Cross-Entropy (BCE) loss for the confidence scores of each and every predicted bounding box. In object detection, the network outputs a confidence score for each class, and the class loss penalizes incorrect predictions.

Dual Focal Loss (DFL) function [16] focuses on hard-to-detect examples. It alleviates the class imbalance issue in classification as well as semantic segmentation. This loss function is inspired by the characteristic of the Focal Loss (FL) function that intensifies the loss for a data point yielding a large difference between the predicted and the actual output. Hence, if a data point is hard-to-classify, due to class imbalance or some other reasons, FL makes the neural network focus more on that as well as similar data points. DFL adopts this idea and improves the performance of FL by enhancing the gradient condition. Note that, in a single-class detection scenario, class imbalance refers to the imbalance between the detection target objects and the background, which is usually considered the negative class.

5.3 TACO Training

This section presents the approach and results obtained with the models trained on the TACO dataset. First, training setup is described, including the parameters and methodologies used. Then, the results obtained from the various models trained on the single-class TACO dataset, and subsequently tested on the UniMiB Trash and PlastOPol datasets are illustrate and discuss.

5.3.1 YOLO setup

In this work only most popular versions of YOLO are considered, implemented by Ultralytics library, i.e., YOLO-v5 [36] and YOLO-v8 [18], 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.

5.3.2 TACO-1 task

In this work, only the most popular versions of YOLO implement implemented by Ultralytics library are considered, i.e., YOLO-v5 [36] and YOLO-v8 [18], focusing on the *tiny* and *small* models.

The training trend is illustrated in Figure 5.3, which shows an examples of YOLO v8 nano training.

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 "relax," 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 previously explained, 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 Table 5.2. 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[7][8] are achieved by a YOLO v5 or v8 nano - YOLO v5x is 20x larger than a nano model. YOLO v5s of this work outperforms the others YOLO v5s [7] [8] 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, particu-

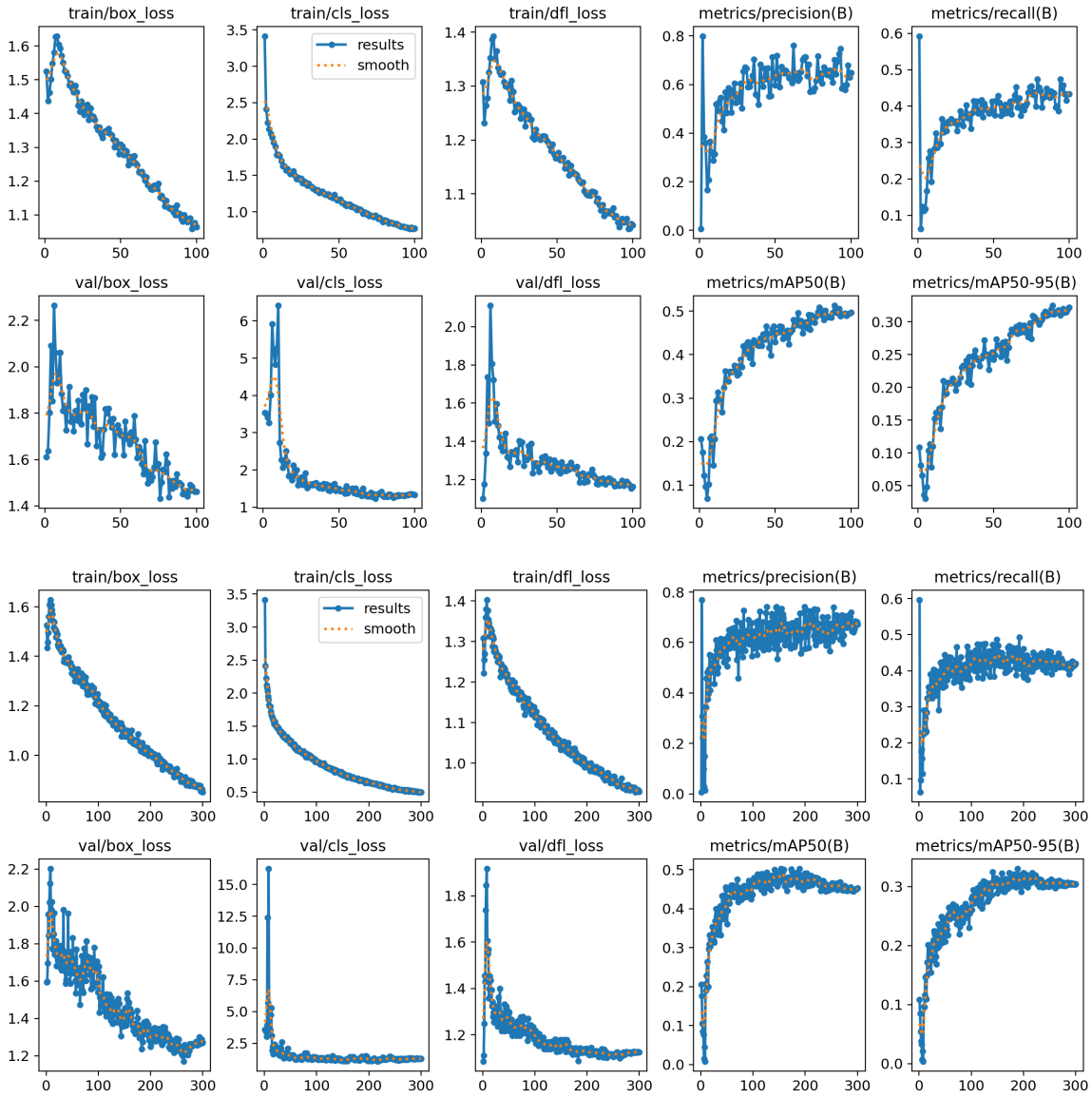

Figure 5.3: Training of YOLO v8 nano model on TACO train and validation set with 100 epochs **(above)** and with 300 epochs **(below)**. Monitoring the trend of the curves on the validation set with the different metrics shows that the model about the hundredth epoch has already achieved the best possible performance.

Table 5.2: Litter detection results on TACO dataset on the TACO-1 task. The best results are highlighted in bold, and only increments greater than 7.0% are reported.

Method	Default thresholds mAP50	mAP50-95	recall	precision	Tuned thresholds mAP50	mAP50-95	recall	precision	Tuned Value Conf	Iou	Size (MB)
RetinaNet [7]	50.6										297.0
Faster R-CNN [7]	51.1										580.0
Mask R-CNN [7]	52.3										491.0
EfficientDet-D0 [7]	32.7										17.0
EfficientDet-D2 [25]	56.8	40.4									35.0
EfficientDet-D5 [7]	42.3										136.0
YOLO-v5s [7]	54.7										15.0
YOLO-v5s [8]	55.0	38.5									15.0†
YOLO-v5x [7]	**63.3**										171.0
YOLO-v5x [8]	61.4	**47.6**									171.0†
YOLO-v5n (this work)	52.2	34.3	47.2	69.2	$62.1^{+\mathbf{9.9\%}}$	$44.5^{+10.2\%}$	46.8	73.4	0.25	0.2	**5.0**
YOLO-v5n6u (this work)	59.5	41.7	50.9	73.8	$66.7^{+7.2\%}$	$\mathbf{50.5}^{+8.8\%}$	52.9	73.3	0.25	0.4	8.3
YOLO-v8n (this work)	51.8	34.7	46.6	68.8	$61.6^{+9.8\%}$	$45.4^{+10.7\%}$	46.7	70.2	0.2	0.2	6.0
YOLO-v5s (this work)	57.8	39.3	50.7	77.3	$65.4^{+7.6\%}$	$48.8^{+9.5\%}$	50.3	**77.7**	0.2	0.5	17.7
YOLO-v5s6u (this work)	62.0	40.7	52.7	77.1	**68.1**	$47.9^{+7.2\%}$	**53.9**	76.1	0.3	0.5	29.6
YOLO-v8s (this work)	56.5	39.6	49.3	74.9	$65.2^{+8.7\%}$	$50.2^{+\mathbf{10.6\%}}$	49.4	75.4	0.3	0.6	21.5

† not declared by the authors, taken from [7]

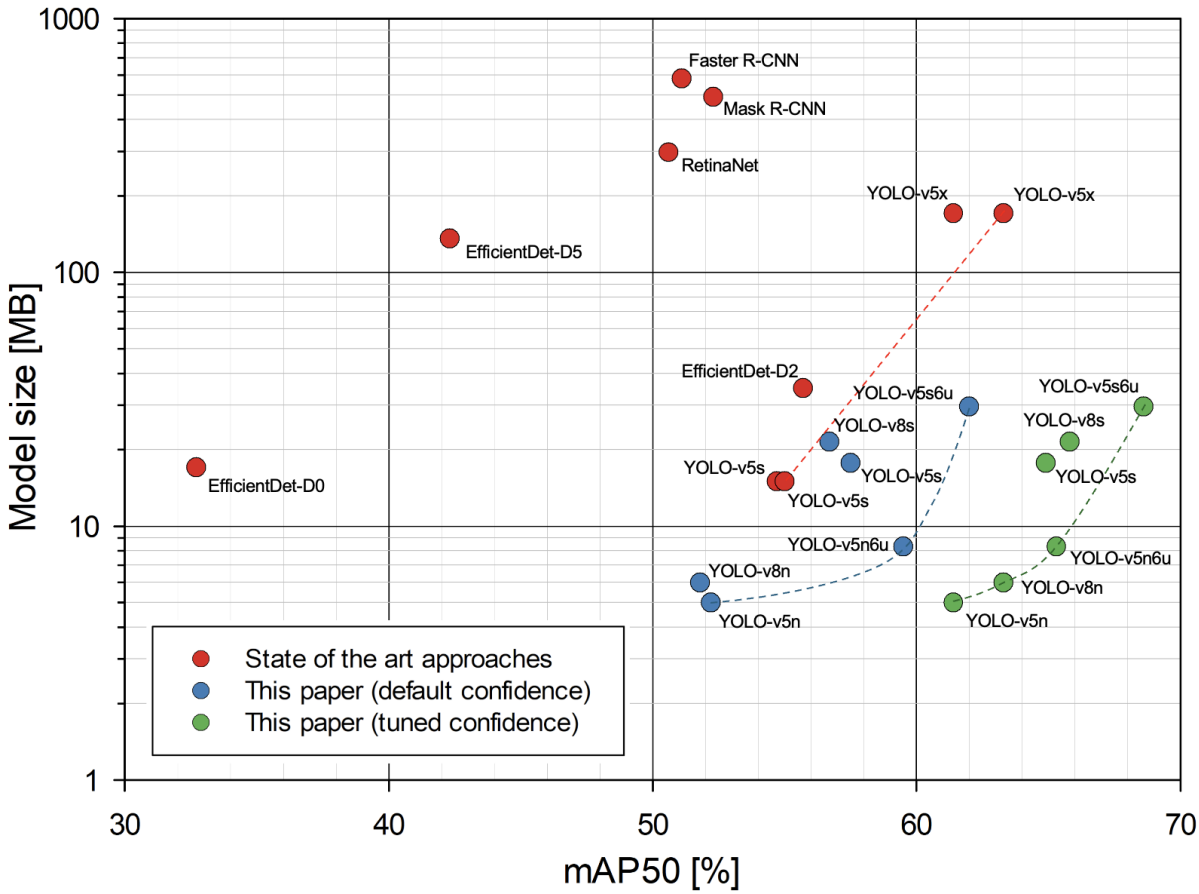

Figure 5.4: Performance comparison on the TACO dataset on the TACO-I task in terms of mAP50 and model size for approaches in the state of the art **(red circles)**, our models with default confidence **(blue circles)**, and our models with tuned confidence **(green circle)**. For each group of models, the corresponding performance Pareto front is reported with a dashed line having the same color.

larly 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, Figure 5.4 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, from Figure 5.5 clearly illustrates their ability to run on devices with low computational capabilities. YOLO v5n6u and

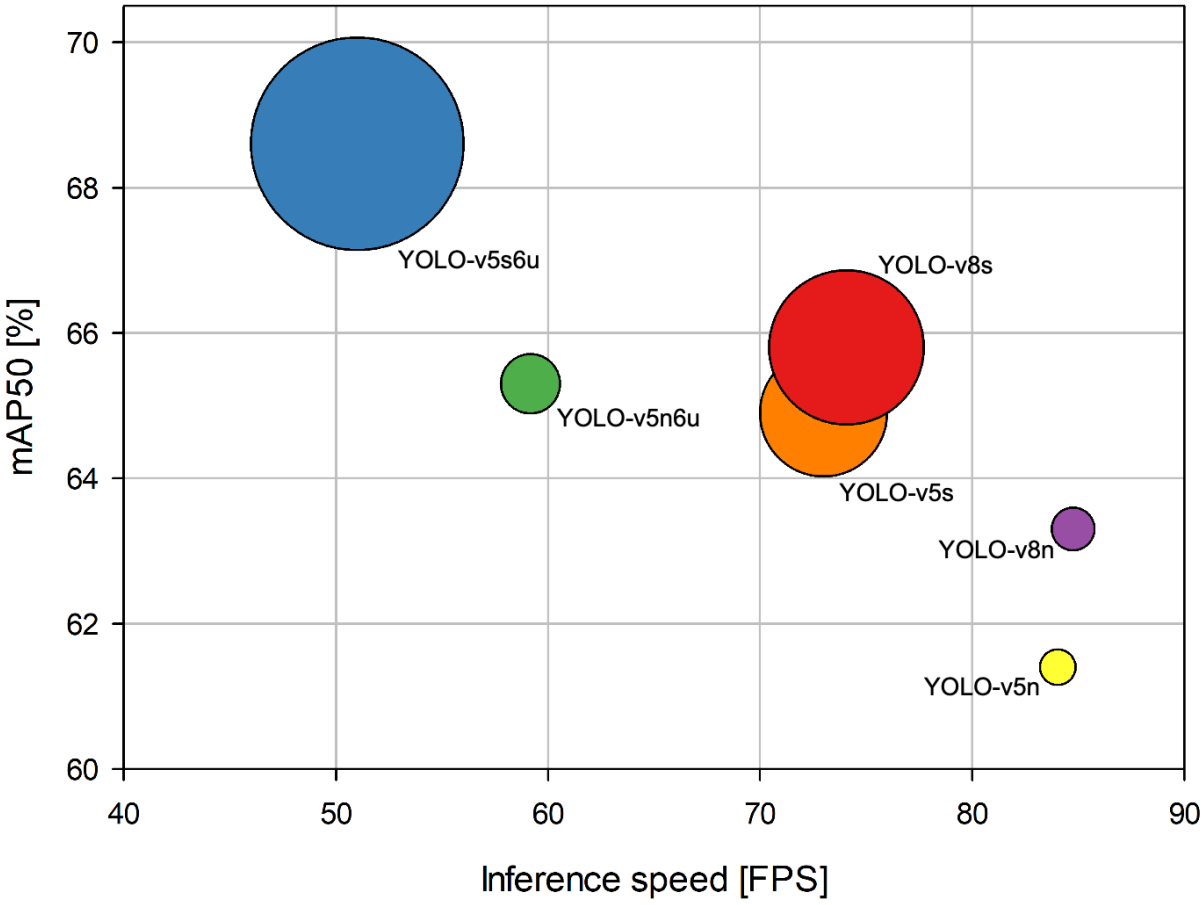

Figure 5.5: Performance comparison on the TACO dataset on the TACO-I task in terms of mAP50, inference speed and model size for our models with tuned confidence.

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.

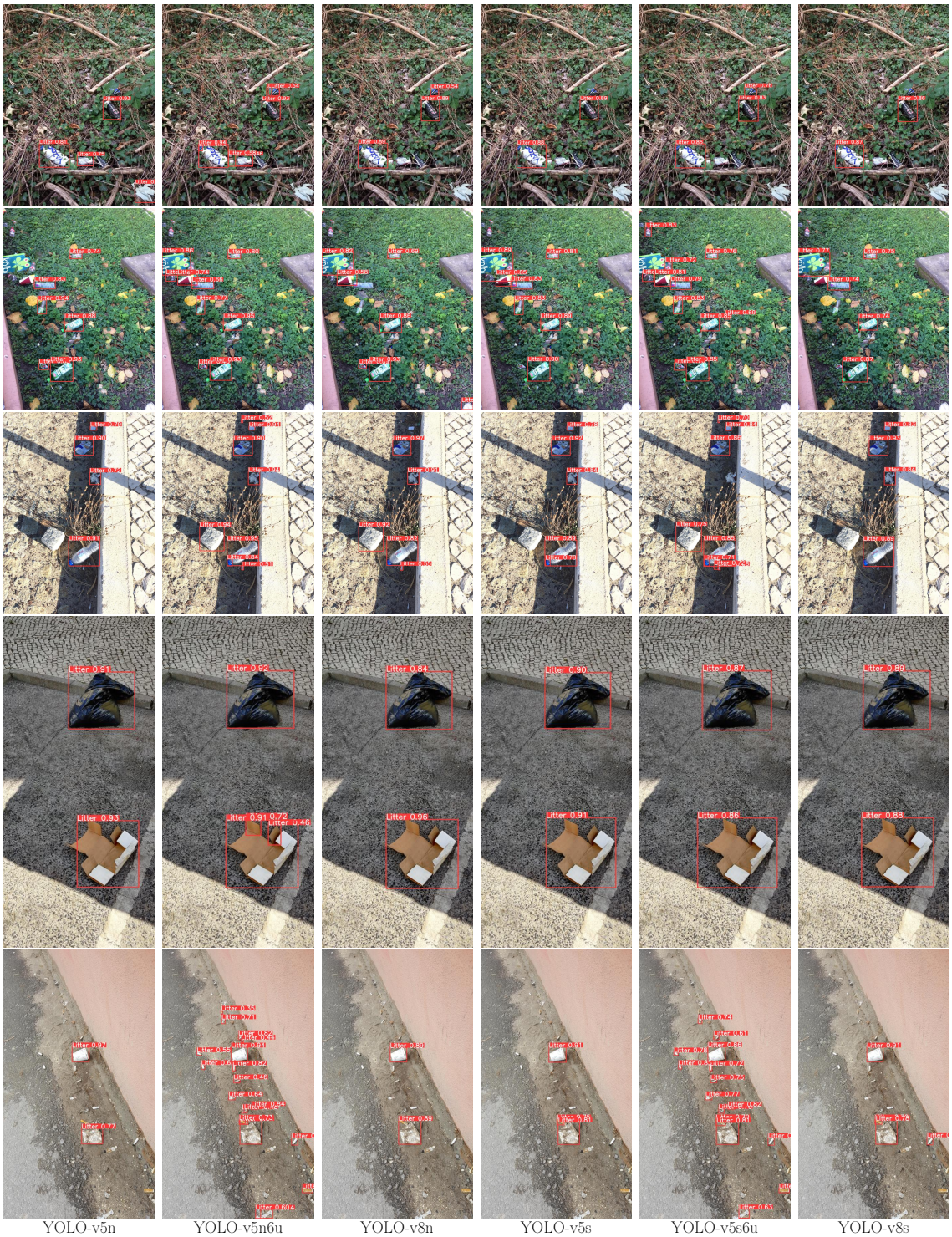

Figure 5.6: Litter detection results of our models on some sample images belonging to the test set of the TACO dataset. Detected litter is annotated with a red bounding box reporting also its detection confidence.

Table 5.3: Results on each fold of the PlastOPol dataset. Following the authors' workflow, every fold is evaluated, and the one where the models generally performed best was selected for subsequent tuning. To determine the best-performing fold, a scoring system was used: 1 point was awarded for each instance where a model achieved a higher mAP50 than others, and 0.5 points for a higher mAP50-95, as detecting objects is considered more important than detecting them with greater precision. Based on this evaluation, the choice narrowed down to the first and third fold.

	YOLO-v5n		YOLO-v5n6u		YOLO-v8n		YOLO-v5s		YOLO-v5s6u		YOLO-v8s		
Method	mAP50	mAP50-95	mAP50	mAP50-95	mAP50	mAP50-95	mAP50	mAP50-95	mAP50	mAP50-95	mAP50	mAP50-95	Total
Fold 1	62.0	43.5	**68.7**	**51.9**	**64.2**	46.4	**59.3**	43.7	66.1	48.7	**63.6**	47.5	**4.5**
Fold 2	61.2	45.4	63.9	48.6	61.7	46.8	59.1	**45.4**	62.6	45.4	62.1	48.7	0.5
Fold 3	64.6	**47.2**	67.6	51.4	63.7	**47.5**	58.4	44.7	**67.6**	**49.3**	62.1	**48.9**	3
Fold 4	58.8	42.5	64.6	48.1	59.7	44.7	58.1	43.3	63.0	44.7	61.2	46.2	0
Fold 5	**64.9**	45.4	67.0	49.4	62.8	46.0	**59.3**	44.1	64.3	45.3	63.5	48.3	2

5.3.3 PlastOPol

This dataset is not used for training new models but rather to assess the quality of the TACO dataset. If models trained on TACO maintain their performance on an external dataset, it increases the likelihood that the trained models will also perform well in real world scenarios. However, if the performance significantly degrades, the approach may need to be reconsidered.

PlastOPol follows an official split of 80% training and 20% test sets. The training set is further divided into five random subsets, with four used for training and one for validation. This procedure allowed the authors to use a 5-fold cross-validation with these folds. Unfortunately, the authors do not specify which fold is used as the validation set: *"[...] we selected the best model on the validation set from the 5 folds and used it to compute the final results on the test set."*

To replicate the workflow described by the PlastOPol authors, the models trained on the TACO dataset were evaluated on each fold of the PlastOPol dataset. Inference was performed on all folds, and the metrics were compared to select the fold with the best performance. The results are presented in Table 5.3 and visualized in Figure 5.7. Fold 1 emerged as the best performing fold, using it to computing the hyperparameter tuning. These tuned parameters were then applied to the test set. While this choice is significant, it is not overly critical: unlike the original PlastOPol approach, where one of five different models was selected based on test set results, here only the hyperparameters are tuned, and the underlying models remain unchanged.

The authors of [8] have reported results on fold 3, which is actually part of the training set. Essentially, they presented the metrics from the best-performing validation fold rather than from the test set. Due to this discrepancy, their results are excluded from Table 5.4, which displays the results obtained in this work.

Considering that the models in this work are trained on TACO dataset rather than the PlastOPol dataset, the results are moderately good. The performance of YOLO v5n6u (this work) is not significantly different from that of YOLO v5s in the study by Cordova et al., with a gap of approximately 9% in terms of mAP50 and only 5% in mAP50-95.

This indicates that the TACO dataset generalizes well, being able to train a model that is capable of detect litter outside itself. Another positive finding is that there is no considerable negative gap between performance on the validation and test sets, suggesting that the data sets are realistic and the models have learned. While results may vary from fold to fold in the validation set, or the test set may experience some variability,

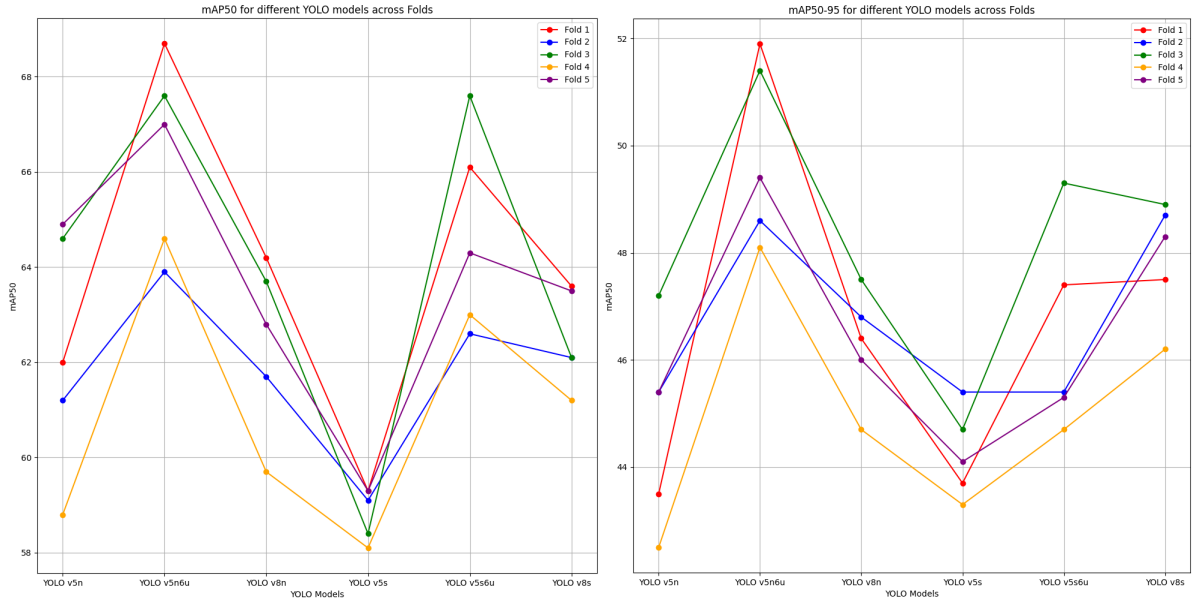

Figure 5.7: The plots based on Table 5.3 illustrate mAP50 **(right)** and mAP50-95 **(left)** for various YOLO models across each fold of the PlastOPol dataset. Folds 1 and 3 stand out due to their consistently above-average performance: Fold 1 exhibits stronger results in terms of mAP50, while Fold 3 shows better performance in mAP50-95. Considering the overall results, the first fold was selected for hyperparameter tuning, as it demonstrated higher mAP50 values, which is considered more critical for object detection

performance consistently remains within a reasonable range.

Both with default parameters and after thresholds tuning, the differences between the nano and small models are minimal compared to the results obtained on TACO. Only YOLO v5s6u exhibits a greater Recall, but this advantage comes at the cost of a lower Precision, with only a 5% difference compared to YOLO v5n.

5.3.4 UniMiB Trash Test

To further test the trained models following the PlastOPol experiment, it was decided to use the UniMiB Trash dataset, which allow a testing in realistic and relevant scenarios. The labels for large, medium, small, and micro classes were converted into a single class (litter) to be able to use the models trained on the TACO dataset. Images containing cluster annotations were obviously excluded, as it is unreasonable to expect models trained on TACO or PlastOPol to detect these types of waste.

The entire UniMiB Trash dataset was employed for testing, the same value of Conf and IoU is applied for the testing on TACO. Since an official partition was not defined at the time of testing, utilizing the complete dataset as a large test set for the TACO-trained models was beneficial. This approach provided a valuable opportunity to evaluate model performance on a substantial number of realistic images, which would be impossible to find otherwise. As observed in previous experiments, tuning the thresholds (as indicated in Table 5.2) enhances performance and highlights the differences compared to default parameters in a realistic dataset. It is important to note that performing performance tuning directly on this dataset would be inappropriate, as it would effectively convert it into a validation set, compromising the integrity of the evaluation. In uncontrolled envi-

Table 5.4: Litter detection results on PlastOPol dataset using models trained on TACO dataset. The best results are highlighted in bold, and only increments greater or less than 7.0% are reported.

Method	Default thresholds				Tuned thresholds				Tuned Value		Size
	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

ronments, the ground truth for input images is unavailable, preventing the optimization of hyperparameters.

Table 5.5: Litter detection results on UniMiB Trash dataset. There is still an increase in performance due to confidence tuning. The first place spot is closely contested between YOLO v5n6u and YOLO v5s6u. Nano models are still the ones that benefit most from tuning thresholds.

Method	Default thresholds				Tuned thresholds				Tuned Value		Size
	mAP50	mAP50-95	recall	precision	mAP50	mAP50-95	recall	precision	Conf	Iou	(MB)
YOLO-v5n (this work)	56.0	37.0	48.2	67.4	$63.4^{+7.4\%}$	$46.6^{+9.6\%}$	44.7	$\mathbf{78.4}^{+11.0\%}$	0.25	0.2	**5.0**
YOLO-v5n6u (this work)	63.4	45.1	54.6	72.2	67.9	$\mathbf{52.2}^{+7.1\%}$	53.5	77.0	0.25	0.4	8.3
YOLO-v8n (this work)	54.4	36.8	46.7	67.6	$62.6^{+8.2\%}$	$47.0^{+10.2\%}$	45.0	$75.3^{+7.7\%}$	0.2	0.2	6.0
YOLO-v5s (this work)	57.9	40.3	48.9	73.6	$64.9^{+7.0\%}$	$49.2^{+8.9\%}$	49.7	74.1	0.2	0.5	17.7
YOLO-v5s6u (this work)	63.6	40.9	55.5	73.3	**68.5**	47.0	**56.3**	73.9	0.3	0.5	29.6
YOLO-v8s (this work)	58.6	41.1	50.0	72.4	65.3	50.8	48.2	76.8	0.3	0.6	21.5

The results on the UniMiB Trash dataset, as shown in Table 5.5, are consistent with the previous ones: in scenarios without cluster, litter detection using very small deep learning models is effective. Although these models do not achieve the highest possible performance, they are extremely lightweight and capable in addressing a challenging task. Once again, YOLO v5n6u and YOLO v5s6u outperform their competitors, likely due to their higher input resolution, which helps in detecting smaller objects.

5.4 UniMiB Trash multiclass task

As previously discussed, the PlastOPol dataset contains waste instances annotated under a single class, "Litter". In contrast, the TACO dataset defines 60 categories of waste types, which are further grouped into 28 superclasses. However, TACO contains relatively few images given the number of classes. For context, the Ultralytics YOLO guidelines recommend at least 1,500 images per class and approximately 10,000 instances per class. TACO falls significantly short of this recommendation. Not surprisingly, even the authors of the dataset themselves propose a 10-class classification first, and then a 1-class classification, which in itself is by no means simple.

This work also aims to explore whether the task can be approached in a multiclass context, providing additional information beyond the quantity of waste 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 *cluster* class has been excluded from this approach, as it is not suitable for an object detection task. Detecting litter clusters may require custom metrics, and with the current setup, the models are not equipped to identify such clusters effectively. This version clarifies the purpose and reasoning behind the classification choices while maintaining a formal, scientific tone.

The 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. However, it is important to note that the reported results should be regarded as indicative rather than definitive, as different partitioning strategies can lead to variations in performance metrics. In particular, a more favorable test set could result in artificially inflated performance, depending on the random split. To address this, implementing K-Fold cross-validation would be one of the most rigorous approaches, as it allows for more robust performance evaluation by averaging results across multiple folds. Unfortunately, due to hardware limitations, this could not be accomplished, since Google Colab was used for the experiments.

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 Confidence parameter for threshold tuning, probably due to the ability to predict with higher confidence score. Some training results are shown in Figure 5.8.

Results on test test are shown in table 5.6. Considering that the dataset is incomplete, 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 an higher number of classes, tuning the Confidence and IoU thresholds becomes more challenging. As shown in Figure 5.9, 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, which performs significantly better and demonstrates good performance, likely due to the higher resolution of the input images.

The confusion matrices shown in Figure 5.10 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, 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

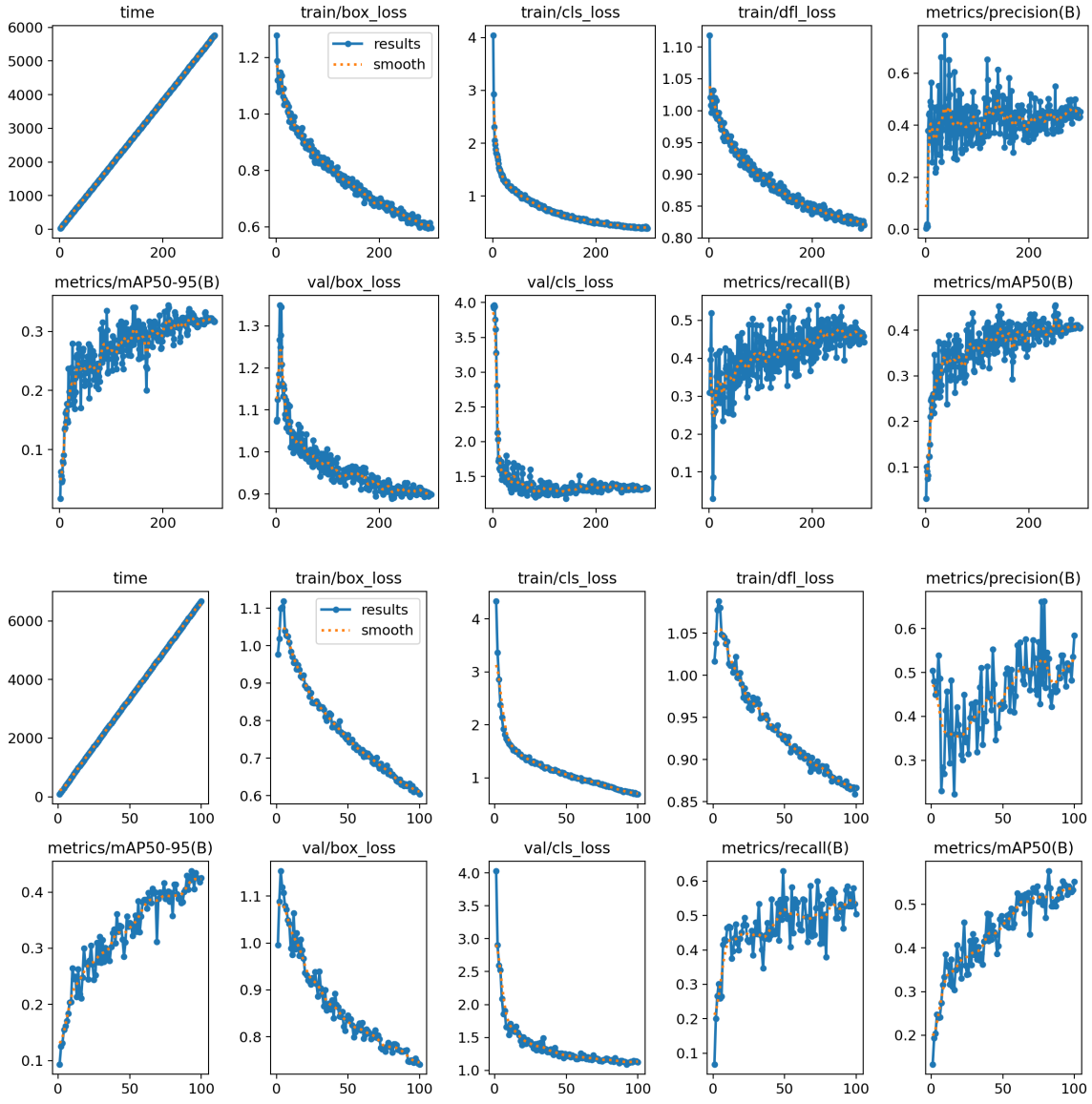

Figure 5.8: **(Above)** Training results with 300 epochs using YOLO v8n on the UniMiB trash datast. The trend on validation set is fairly regular: after the rapid improvement in the first 50 epochs, the model's learning rate slows down, with the class loss curve becoming nearly flat. **(Below)** Training result with 100 epochs using YOLO v5s6u. The computational time per epoch is approximately four times longer than YOLO v8n, mainly due to the higher resolution of the input images. With Google Colab, it is not feasible to exceed 100 epochs, although there appears to be potential for further performance improvements with YOLO v5s6u.

Table 5.6: Detection results on the UniMiB Trash test set, considering all four classes. Best results are highlighted in bold. Threshold tuning is applied on all classes, although ideally, tuning would be performed separately for each class, as they respond differently to threshold adjustments. In this case, while the performance improvement is not as pronounced as on the TACO dataset, there is still an overall enhancement.

		Default thresholds				Tuned thresholds				Tuned Value		Size
Method	Class	mAP50	mAP50-95	recall	precision	mAP50	mAP50-95	recall	precision	Conf	Iou	(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			

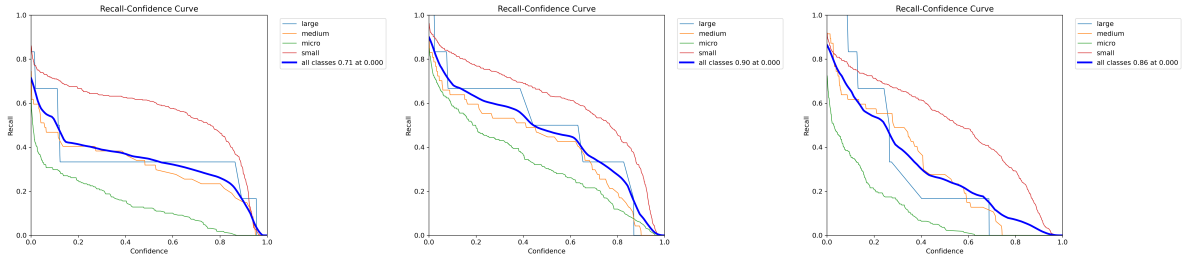

Figure 5.9: Precision-Recall curve of YOLO v8n **(left)**, YOLO v5s6u **(center)** and YOLO v8s **(right)** on UniMiB Trash dataset. In a multiclass task, threshold tuning becomes more challenging because the predicted classes may have different confidence levels, depending on how the model has learned.

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. 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.

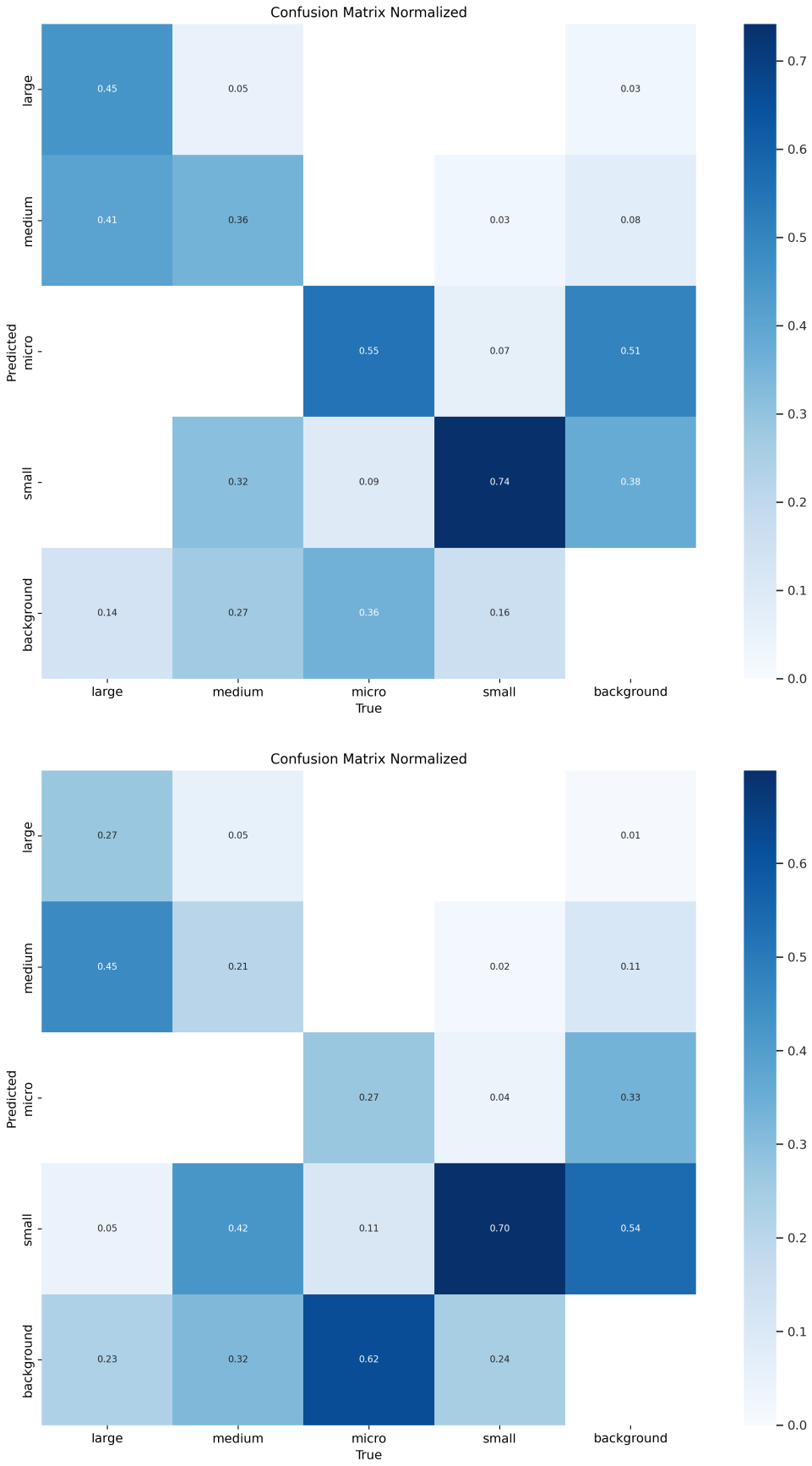

Figure 5.10: Example of normalized Confusion Matrices of YOLO v5s6u **(above)** and YOLO v8s **(below)** on UniMiB Trash test set with 4 class.

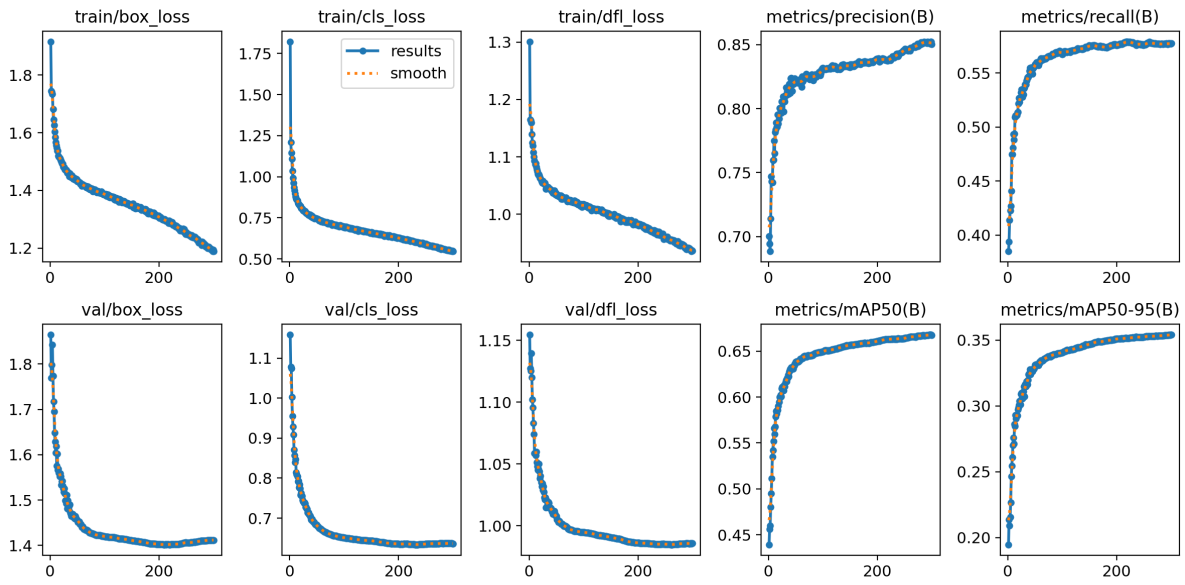

Figure 5.11: YOLO-v8n train on Wider Face dataset. The trend of the loss functions on the validation set indicates that the model has nearly reached its maximum performance.

5.5 Wider Face

Wider Face provides an official split into training, validation and test set. Unfortunately, the ground truths of test set are not available. The computational costs required to train several models with this dataset are very high. Without access to a local machine and relying on Google Colab for initial experimentation, using the entire dataset is a infeasible. Therefore, several initial trials were conducted with an undersampled version of the training set, primarily using YOLO v8n.

As described in Chapter 3, Wider Face categorizes images into three levels of difficulty: easy, medium, and hard. The validation set can be partitioned accordingly, so the hard subset is used for validation, while the easy and medium subsets are used as the test set. The training parameters are the same as those used for the TACO dataset, but this dataset requires significantly more training epochs, at least 300.

The results are primarily compared with those in [41], the authors of the Wider Face dataset. Although the performance of the YOLO model could also be compared with [27], but this comparison is not fair: the authors conducted an in-depth study of the problem and implemented significant changes, which differ from the scope of this work, where such changes are considered secondary. Delong Qi et al. made substantial modifications to the architectures of the YOLO v5 models and tuning all hyperparameters for models training. Since this is not the aim of this work, it is not possible to focus on this part as them: the goal is to achieve decent performance, but without reaching to exceed the state-of-art models. Moreover, it remains unclear how the validation sets were used, whether tuning was performed, and what metrics were employed. While it is assumed that mAP50 was used, only a general reference to "mAP" is reported. The mAP50-95 metric is excluded due to the extremely high values obtained.

The results are presented in table 5.7.

The YOLO model obviously outperforms the models mentioned in [41], which are

Table 5.7: Face detection results on each Wider Face validation set. Also the performance of the YOLO World models, both with the standard and custom two-stage methods, are reported in the table. The performance increase resulting from threshold tuning is significant, particularly for YOLO v8n. In contrast, does not experience the same level of improvement, as it generally predicts with low confidence scores. Reducing the IoU threshold using the default method does help improve performance, but in terms to discard false positive. The recall of the models slighlty decrease (1-2% in mAP50). Only increments greater than 7.0% are reported.

Method	Set	Default thresholds		Tuned thresholds		Tuned Value	
		mAP50	mAP50-95	mAP50	mAP50-95	Conf	Iou
ACF [41]	Easy	69.5					
Faceness [41]	Easy	71.3					
Multiscale Cascade CNN [41]	Easy	71.1					
Two-Stage CNN [41]	Easy	65.7					
YOLO-v8n (this work)	Easy	75.7	44.1	81.2	$52.7^{8.6\%}$	0.2	0.2
YOLO-v8s (this work)	Easy	79.4	47.2	**83.6**	$\mathbf{54.6}^{7.4\%}$	0.2	0.5
YOLO World S zero shot (1 class)	Easy	22.6	9.5	$35.2^{12.6\%}$	14.6	def	0.01
YOLO World S zero shot (two stage)	Easy	61.8	31.7			def	def
YOLO World X zero shot (1 class)	Easy	33.9	17.0	$48.3^{14.4\%}$	$24.5^{7.5\%}$	def	0.01
YOLO World X zero shot (two stage)	Easy	66.0	39.9			def	def
ACF [41]	Medium	58.8					
Faceness [41]	Medium	60.4					
Multiscale Cascade CNN [41]	Medium	63.6					
Two-Stage CNN [41]	Medium	58.9					
YOLO-v8n (this work)	Medium	61.5	34.2	$72.0^{10.5\%}$	$45.1^{10.9\%}$	0.2	0.2
YOLO-v8s (this work)	Medium	66.6	37.7	$\mathbf{74.6}^{8.0\%}$	$\mathbf{47.0}^{9.3\%}$	0.2	0.5
YOLO World S zero shot (1 class)	Medium	17.4	6.9	$30.3^{12.9\%}$	11.5	def	0.01
YOLO World S zero shot (two stage)	Medium	52.8	26.8			def	def
YOLO World X zero shot (1 class)	Medium	27.7	13.2	$42.9^{15.2\%}$	$20.7^{7.5\%}$	def	0.01
YOLO World X zero shot (two stage)	Medium	63.8	37.2			def	def
ACF [41]	Hard	29.0					
Faceness [41]	Hard	31.5					
Multiscale Cascade CNN [41]	Hard	40.0					
Two-Stage CNN [41]	Hard	58.9					
YOLO-v8n (this work) †	*Hard*	*66.9*	*35.5*	$74.9^{8.0\%}$	$44.6^{9.1\%}$	*0.2*	*0.2*
YOLO-v8s (this work) †	*Hard*	*72.3*	*39.4*	*78.0*	$47.2^{7.8\%}$	*0.2*	*0.5*
YOLO World S zero shot (1 class)	Hard	13.8	5.5	$31.8^{18.0\%}$	$12.6^{7.1\%}$	def	0.01
YOLO World S zero shot (two stage)	Hard	**54.0**	30.0			def	def
YOLO World X zero shot (1 class)	Hard	24.5	11.7	$44.6^{\%}$	$22.0^{\%}$	def	0.01
YOLO World X zero shot (two stage)	Hard	50.4	**31.7**			def	def

† Should not be considered a true test set, but rather a validation set, as threshold tuning is performed.

outdated and no longer in use. As before, tuning of confidence and IoU is critical for enhancing performance in terms of mAP50 and mAP50-95. Although Recall and Precision are not reported, tuning was consistently aimed at improving overall performance without sacrificing Recall.

The gap between YOLO v8n and YOLO v8s is not substantial; once threshold tuning is complete, the difference in mAP50 and mAP50-95 ranges from approximately $\sim [2\% - 4\%]$. The performance of YOLO v8n and YOLO v8s on the hard set is shown in italics because the hard set is treated as a validation set for hyperparameter tuning, rather than a true test set like the medium and easy sets. It is unclear how the other authors addressed this issue, as ground truth data are not available. Additionally, [27] only reports results on these validation sets, where the models may have been tuned to achieve optimal performance.

YOLO World in a zero-shot scenario with a two-stage pipeline demonstrates capabilities comparable to YOLO v8n and v8s, despite not being trained on the source dataset. This is a remarkable achievement, considering that YOLO World is pre-trained on several, none of which include the face class. Furthermore, with the default hyperparameters, YOLO World X using the two-stage pipeline outperforms YOLO v8n on the medium set. Additionally, YOLO World in a zero-shot scenario without the two-stage method shows lower performance compared to its rivals. This performance can be improved by decreasing the IoU threshold; it was more of an intuition, not a true tuning in a range of possibilities.

5.6 License Plates

As explained in Chapter 3, the dataset obtained is a combination of various available datasets, so there is no other research for comparison. The training parameters differ slightly because this task is fundamentally distinct from the previous ones; it is essential to train models to enhance their robustness against perspective distortions. For data augmentation, the *degrees* parameter is increased from 10 to 20 to enhance the rotation of images. The *translate* parameter adjusts the image horizontally and vertically by a fraction of the image size, while the *shear* parameter simulates the effect of objects viewed from different angles. Additionally, the *perspective* parameter applies random perspective transformations to the image. The number of epochs is set to 100; the motivation for not increasing the number of epochs is illustrated in 5.12

The workflow followed is consistent: the model is trained using the training set and validated on the validation set during training. After completing the training, threshold tuning is performed on the validation set, and the best hyperparameters are selected for testing the models on the test set.

As shown in table 5.8, the only model that performs below expectations is YOLO v5s6u. Despite training the model three different times, the results are significantly lower than its competitors. As already discussed in Chapter 4, here the performance of YOLO World in two-stage mode is very high. While it is true that this dataset is not as challenging as Wider Face, the strategy of first detecting the cars, which are usually larger in the images, and then detecting the license plates, proves to be very effective.

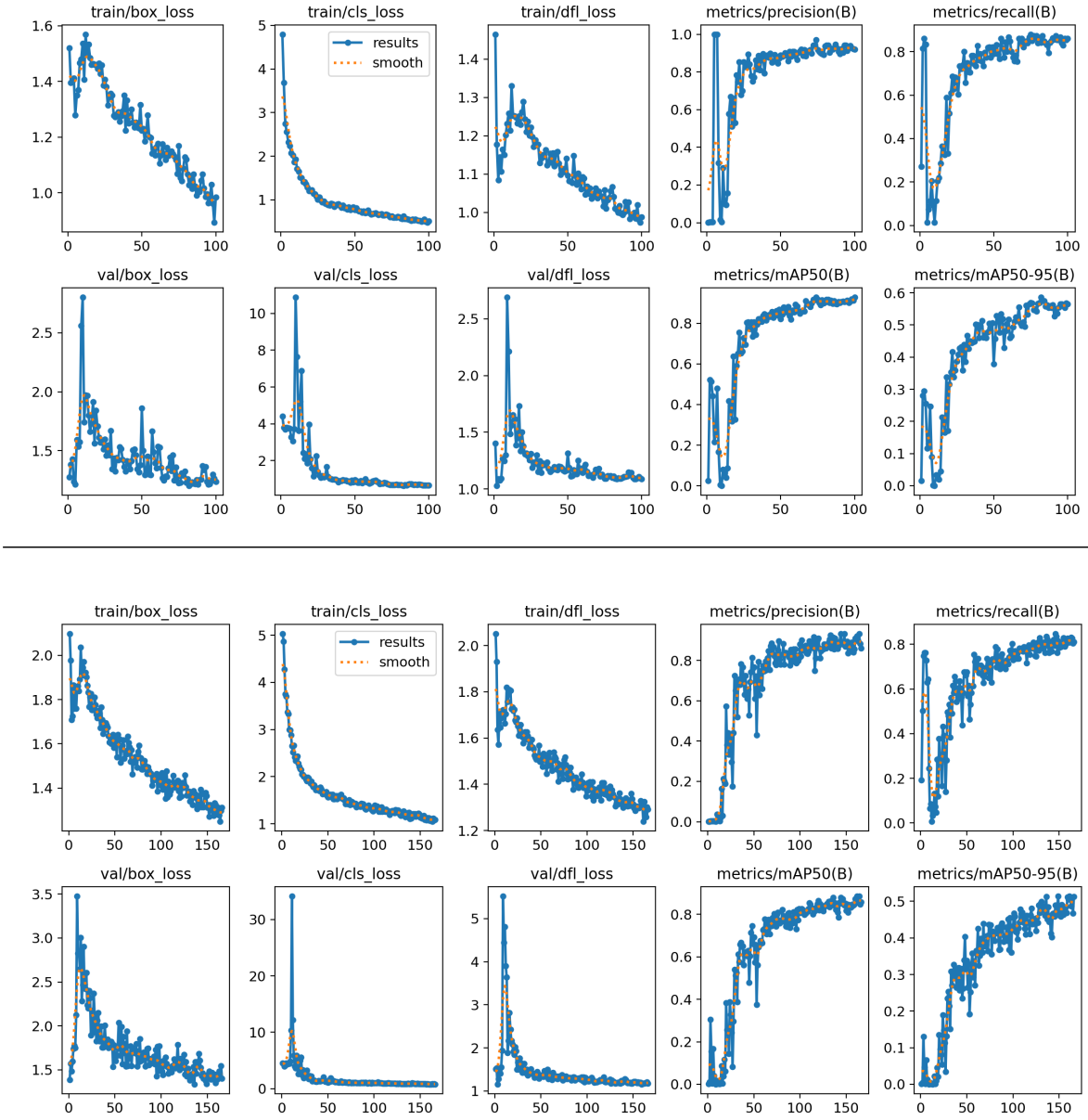

Figure 5.12: The Precision-Recall curve for YOLO-v8n is displayed for training with 100 epochs **(above)** and 200 epochs **(below)**. The model trained for 100 epochs performs slightly better. It is often observed that, when using the YOLO trainer, increasing the number of epochs does not necessarily lead to improved performance; rather, it may simply indicate a tendency for the trainer to "relax" while achieving similar results.

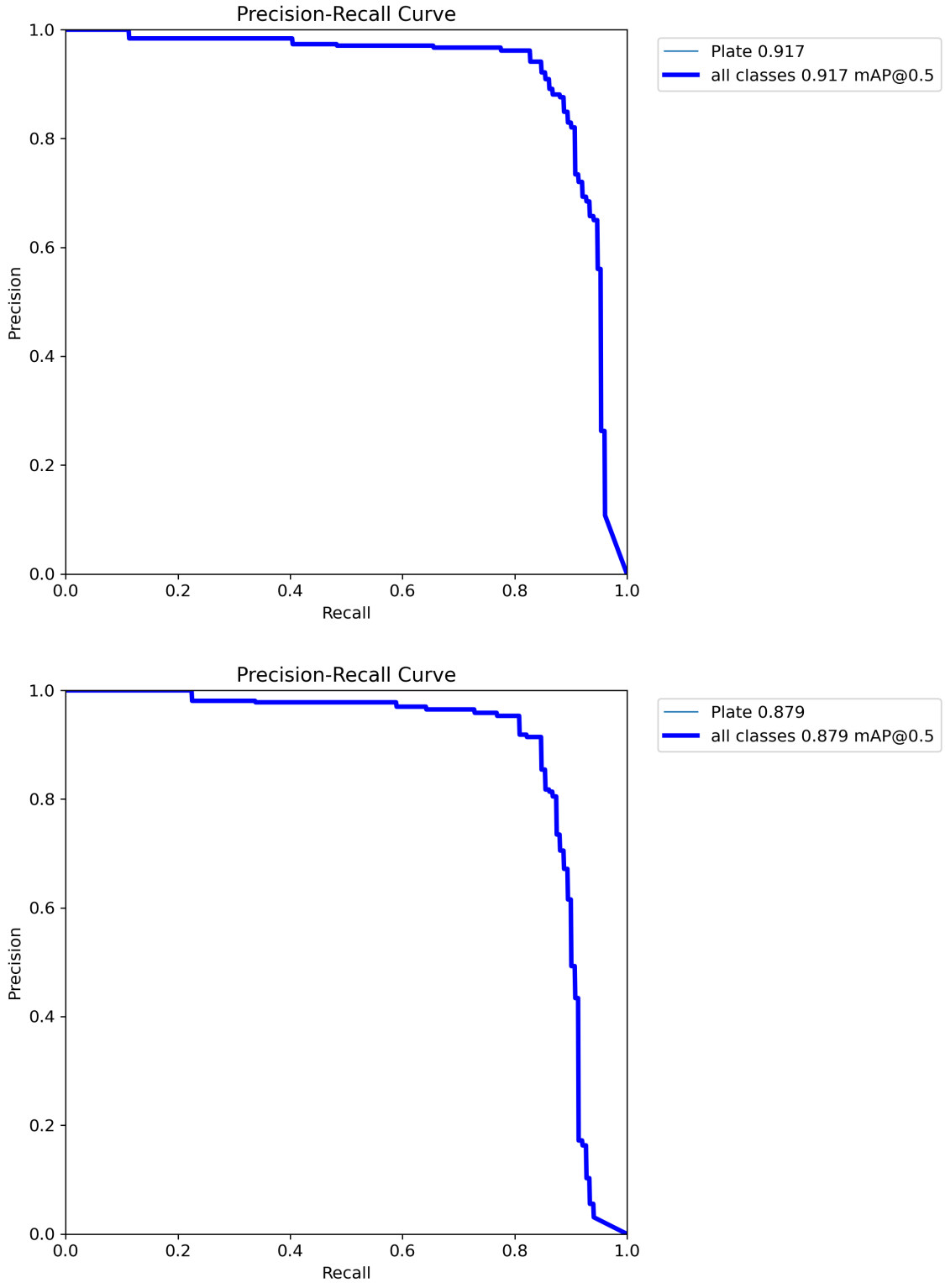

Figure 5.13: YOLO-v8n Precision-Recall curve is presented for training with 100 epochs **(above)** and 200 epochs **(below)**. The model trained with fewer epochs performs better. The mAP50 metric aids in understanding the differences (even so small) between the curves, being the calculation of the area under the curve.

Table 5.8: License Plate detection results on the test set of builded dataset. Threshold tuning helps improve performance, though not as much as with the previous dataset. However, since the baseline performance is already high, achieving even a small increase in mAP50 and mAP50-95, along with an improvement in Recall, is a good outcome. The most substantial improvement is seen in mAP50-95, indicating that the models are better at predicting objects with more accurate bounding boxes. Only increments greater than 7.0% are reported

	Default thresholds				Tuned thresholds				Tuned Value	
Method	mAP50	mAP50-95	recall	precision	mAP50	mAP50-95	recall	precision	Conf	Iou
YOLO-v5n (this work)	84.3	45.5	80.7	85.0	86.4	50.8	83.3	85.9	0.2	0.4
YOLO-v5n6u (this work)	86.1	45.6	83.6	82.9	88.5	49.9	85.7	85.5	0.1	0.4
YOLO-v8n (this work)	88.4	52.7	85.7	89.2	90.0	**57.2**	85.7	89.7	0.2	0.5
YOLO-v5s (this work)	88.3	52.1	85.8	84.7	89.9	56.9	**88.6**	84.2	0.1	0.4
YOLO-v5s6u (this work)	71.7	33.1	68.7	76.2	76.7	38.1	74.4	76.9	0.2	0.2
YOLO-v8s (this work)	87.3	48.5	83.9	85.2	89.2	53.4	87.9	85.5	0.1	0.3
YOLO World S zero shot (1 class)	40.5	16.5	43.5	48.1	$47.9^{7.2\%}$	19.5	58.3	46.3	def	0.01
YOLO World X zero shot (1 class)	47.4	21.2	63.1	47.8	$57.0^{9.6}$	25.6	64.9	50.4	def	0.01
YOLO World S zero shot (4 classes)	43.2	17.0	66.7	25.6	45.0	16.8	56.5	40.2	def	0.01
YOLO World X zero shot (4 classes)	60.1	26.8	68.9	50.6	61.7	27.1	74.9	55.7	def	0.01
YOLO World S zero shot (two stage)	**90.3**	56.0							def	def
YOLO World X zero shot (two stage)	87.3	52.8							def	def

5.7 Three Model Three task vs One Model One Task

Is it better to use a single model to detect the Litter, Face, and License Plate classes, or is it necessary to use three different models, each trained for a single task? The answer to this question can be taken from the Table 5.9. To achieve comparable performance on the mixed datasets, it is necessary to train the models using the entire Wider Face dataset, specifically the Triple Large and Triple Trash Large subsets. Among these, the performance is about the same, with the latter gaining almost 7% in terms of Recall for the Litter class, due to the addition of PlastOPol and UniMiB Trash in the training set.

The comparison between the three YOLO v8n models and a single YOLO v8n trained on the Triple Trash Large dataset generally shows a slight disadvantage for the the single model approach, with an average recall loss of around 3% for each class. However, it's important to note that using a single model offers significant advantages in terms of computational efficiency, requiring only one prediction instead of three, and reduces the storage space required (three times less). The same considerations apply to YOLO v8s, where the performance gap is smaller, particularly in terms of recall, with the Triple Trash Large model nearly matching the three-model setup.

However, a reasonable comparison can be made between three YOLO v8n models and a single YOLO v8s model used for the three tasks. If $0(n)$ represents the computation time of a single YOLO v8n model, then three nano models will have a computation time of $O(3 \cdot n)$ while, on average, one YOLO v8s model will have an inference time of $2 \cdot n$. In terms of model size, using three nano models will occupy 24.9 MB, compared to 21.5 MB for the YOLO v8s. Regarding performance, measured in mAP50, mAP50-95, Recall, and Precision, a YOLO v8s model trained on the Triple Trash Large dataset generally outperforms the approach using three YOLO v8n models. Therefore, the choice of using a single, slightly larger model, such as the YOLO v8s, rather than three individual models, is the optimal solution.

It should also be noted that tuning the Confidence and IoU parameters currently results in a higher performance increase for single models, as the thresholds can be set according to the specific class. In contrast, with single models that uses multiclass predic-

Table 5.9: Results on the respective test sets of the TACO, Wider Face (validation easy + validation medium), and License Plate datasets are presented by comparing models trained on individual datasets against models trained on mixed datasets. In the training set section, *single* refers to a model trained only on the training set of the dataset used for that class. *Triple Light* and *Triple Large* indicate that the training set consists of TACO, Wider Face, and License Plate datasets, with Wider Face either subsampled in the former and used in its entirety in the latter. *Triple Trash Large* is similar to *Triple Large*, but includes the PlastOPol and UniMiB Trash datasets in the training set. The results shown in bold represent the best values obtained with the mixed datasets, comparing these values against those from models trained individually on their respective datasets and highlighting the increase or decrease in performance.

Class	Method	Training set	mAP50	mAP50-95	recall	precision Conf	Iou	
Litter	YOLO-v8n	Single	61.6	45.4	46.7	70.2	0.2	0.2
		Triple Light	59.8	42.8	42.2	72.2	0.25	0.4
		Triple Large	60.4	42.6	40.1	75.7	0.2	0.4
		Triple Trash Large	**61.5**$^{-0.1}$	**45.5**$^{+0.1}$	**43.7**$^{-3.0}$	**76.6**$^{+6.4}$	0.2	0.4
	YOLO-v8s	Single	65.2	50.2	49.4	75.4	0.3	0.6
		Triple Light	57.9	42.8	43.8	65.7	0.2	0.5
		Triple Large	61.5	45.3	41.8	**76.4**$^{+1.0}$	0.2	0.3
		Triple Trash Large	**63.0**$^{-2.2}$	**48.0**$^{-2.2}$	**49.2**$^{-0.2}$	68.3	0.2	0.5
Face	YOLO-v8n	Single	75.3	47.9	57.5	87.7	0.2	0.5
		Triple Light	71.2	44.4	51.6	85.3	0.25	0.4
		Triple Large	**74.9**$^{-0.4}$	**47.9**	**56.7**$^{-0.8}$	88.2	0.2	0.4
		Triple Trash Large	74.6	47.6	54.0	**91.3**$^{+3.6}$	0.2	0.4
	YOLO-v8s	Single	77.8	49.7	61.7	88.5	0.2	0.5
		Triple Light	74.2	46.2	55.4	88.3	0.2	0.5
		Triple Large	**77.7**$^{-0.1}$	49.7	60.1	**90.7**$^{+2.2}$	0.2	0.3
		Triple Trash Large	77.5	**49.8**$^{+0.1}$	**61.3**$^{-0.4}$	88.0	0.2	0.5
License Plate	YOLO-v8n	Single	90.0	57.2	85.7	89.7	0.2	0.5
		Triple Light	88.5	57.1	**83.3**$^{+2.4}$	87.5	0.25	0.4
		Triple Large	89.1	58.5	82.7	**88.0**$^{-1.7}$	0.2	0.4
		Triple Trash Large	**89.7**$^{-0.3}$	**58.8**$^{+1.6}$	82.7	86.7	0.2	0.4
	YOLO-v8s	Single	89.2	53.4	87.9	85.5	0.1	0.3
		Triple Light	90.9	**63.3**$^{+9.9}$	87.5	82.6	0.2	0.5
		Triple Large	**92.5**$^{+3.3}$	61.4	**89.3**$^{+1.4}$	**88.8**$^{+3.3}$	0.2	0.3
		Triple Trash Large	91.8	62.7	88.7	86.1	0.25	0.5

tion, it is only possible to establish unique Confidence and IoU thresholds for all classes. This limitation may hinder the ability to achieve maximum performance, as different classes can be predicted with varying confidence levels depending on the inherent difficulty of the prediction and their representation in the dataset. It is assumed that, by enabling the ability to set IoU and Confidence thresholds on a class-by-class basis, overall performance can still improve, even slightly.

Chapter 6

Conclusions and Future Developments

6.1 Problem Addressed

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. 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.

It is difficult to assert which task has been fully resolved: many challenges have been tackled in this dynamic and innovative project, where the demands are very high, and current technologies are not yet fully ready to meet all the project's requirements. However, from a research perspective, it can be said that some small progress has been made, having analysed the state of the art and having laid the foundations for future implementations. Certainly, the task that could be considered within the reach of current techniques and available data is the detection of litter using one class, the Litter class as in TACO-1 task. The tuning of NMS parameters has led to significant improvements, and the available datasets, although individually insufficient, can be used together to train more robust models for detecting litter in the wild. In particular, the introduction of the UniMiB Trash dataset is highly valuable for incorporating very realistic scenarios. This task can be considered resolved and within reach of even the lighter YOLO models.

6.2 Methods Proposed

This work introduces novel methodologies to address the challenge of detecting litter in the wild and censoring sensitive information, particularly through the tuning of confidence and IoU thresholds in the Non-Maximum Suppression algorithm, increasing model performance to its limits. This method deserves considerable attention in all tasks involving YOLO and any other Object Detector that utilizes it, as it enables enhanced model performance in real-world scenarios without introducing modifications or complex new concepts, but rather by fully exploiting existing capabilities. The approach developed with YOLO World is extremely interesting, as it allows for model use without needing

the re-train on a custom dataset. It proves to be remarkably effective for tasks involving license plate and face classes, although unfortunately, it is not equally beneficial for the litter class.

6.3 Experiments and Results

The experiments conducted on the TACO dataset, training and testing the models, along with the PlastOPol and UniMiB datasets used as test sets for the models developed on the first dataset, demonstrate that even the lightest object detectors can effectively perform waste detection using one class. The TACO dataset generalizes quite well, allowing for the creation of models that operate effectively outside of the dataset itself, achieving good performance on both the PlastOPol and UniMiB Trash datasets.

Tuning the Conf and IoU parameters enables a YOLO Nano to nearly match the performance of a YOLO v5x from previous studies in terms of mAP50, and this is an outstanding achievement.

Regarding the 4-class task on UniMiB Trash, the obtained results indicates that the chosen path may be promising and warrant further exploration: the dataset must be expanded further and annotations must continue to be carefully defined and rechecked. The higher performance of the YOLO v5 models using images with a resolution of 1280 pixels indicates that it is necessary to evaluate the increase in resolution leads to an overall increase in performance, even if it focuses mainly on the detection of smaller objects that usually belong to the micro class. However, it is important to note that objects of all classes may appear on different size scales.

In the task of face detection, the Wider Face dataset is comprehensive and truly challenging. Since it is not the primary focus of this work, we did not concentrate extensively on this task; however, the results obtained are positive and confirm the ability of tuning the parameters involved in Non-Maximum Suppression to enhance the performance of all models.

Regarding the detection of vehicle plates, unfortunately the available datasets in the state of the art are not optimal. The achieved performance with the built dataset are high and we have tried to create the most complete and challenging dataset possible, so the performance is satisfactory. Unfortunately, due to privacy regulations, it is not a given to collect a dataset that allows to make up for the shortcomings. However, the high performance of YOLO World could fill this gap by not requiring a complete training dataset, but by detecting the plates using the two-stage prediction method.

Finally, the results obtained by comparing the individual models trained on the individual datasets against the models trained simultaneously on all three classes are satisfactory. It is essential to consider not only metrics like mAP50 and similar measures but also inference time and resource utilization. Using a single YOLO v8s model trained on a dataset that includes litter, face, and license plate, rather than three different YOLO v8n models each trained on a specific dataset, is much more convenient. This approach yields similar detection performance while providing savings in terms of inference time and resource utilization. Incorporating all the previously mentioned datasets, particularly the addition of PlastOPol and UniMiB Trash to the training set instead of relying solely on the TACO dataset, proves to be a winning strategy. It enhances the models' performance in terms of recall for the litter class, which is fundamental and initially showed the most significant decrease in performance.

Undoubtedly, the greater availability of hardware for greater computational capabilities would enable more and deeper experiments, which sometimes limited experiments, particularly with larger datasets such as Wider Face.

6.4 Future Developments

Here are the future developments that are worth exploring. There are undoubtedly many ways to consider, but only some of them can be pursued. The intention is to highlight what is deemed crucial for the success of this project in both the short and long term

6.4.1 YOLO World with Two-Stage prediction

YOLO World with a two-stage prediction is an interesting method that should be considered in this project. It achieves good performance on the license plate and face classes, but a different strategy must be adopted for the litter class. One possible way to explore is modifying the architecture of YOLO World, to allow the training of the model on the litter class without update the pre-trained model weights. This operation requires non-trivial skills, due to the changes in architecture, and could demand significant time. While it is not believed that the two-stage prediction strategy would be useful for litter detection. However, it would still be interesting to try to detect the generic Litter class, and then predict a several specific classes (based either on type or size) on the cropped images.

6.4.2 Non-Maximum Suppression tuning

The strategy of tuning the Confidence and Intersection Over Union parameters in the Non-Maximum Suppression algorithm is one of the successes of this thesis. The tuning is performed on the validation set once the training of the model is completed. It would be interesting to conduct an in-depth analysis of the results obtained by already setting the confidence parameter to higher values at this stage, in order to understand whether the learning of the model is already beneficial or not. This task is not difficult to perform, but may be limited by the capacity of the available hardware, as it involves the different datasets in this research. One could therefore start with the TACO dataset, as with the tuning of the NMS, and then analyse the results and, if necessary, apply the method to the other datasets as well.

6.4.3 UniMiB Trash

The dataset needs to be expanded by collecting more images and carefully analyzing the dilemma posed by the cluster class. How many images are necessary? 2,000? 10,000? 100,000? The answer is straightforward yet indefinite: as many as possible. Certainly, collecting and, especially, annotating this type of image is quite resource-intensive and challenging. Creating a heterogeneous and realistic dataset is an effort that can only be achieved through collective effort; a single researcher cannot fulfill this task alone. The collected images must meet the initial requirements, and above all, the annotations must consistently adhere to high quality standards. It is particularly essential to focus on gathering images of large waste items (medium and large classes) and on piles of waste (cluster class), as those containing small waste items are relatively few. Intermediate

training sessions should be conducted to assess performance trends, and an official test set must be defined that is representative and serves as a benchmark for the different methods used for detection. Finally, an initial version of this dataset should be published to contribute to scientific research and draw attention to a task that seems to have been neglected. The availability of complete and quality data, the greatest lack in this field, may prompt many researchers to reconsider the litter detection problem.

Bibliography

[1] Siddharth Agrawal. *Global License Plate Dataset*. 2024. arXiv: 2405.10949 [cs.CV]. URL: https://arxiv.org/abs/2405.10949.

[2] Luciano Baresi et al. "COBOL: COmmunity-Based Organized Littering". In: ().

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV]. URL: https://arxiv.org/abs/2004.10934.

[4] Alexander Buslaev et al. "Albumentations: Fast and Flexible Image Augmentations". In: *Information* 11.2 (2020). ISSN: 2078-2489. DOI: 10.3390/info11020125. URL: https://www.mdpi.com/2078-2489/11/2/125.

[5] *Car License Plates Dataset*. URL: https://makeml.app/datasets/cars-license-plates.

[6] Tianheng Cheng et al. *YOLO-World: Real-Time Open-Vocabulary Object Detection*. 2024. arXiv: 2401.17270 [cs.CV]. URL: https://arxiv.org/abs/2401.17270.

[7] Manuel Córdova et al. "Litter Detection with Deep Learning: A Comparative Study". In: *Sensors* 22.2 (2022). ISSN: 1424-8220. DOI: 10.3390/s22020548. URL: https://www.mdpi.com/1424-8220/22/2/548.

[8] Dhrubajyoti Das et al. "Outdoor Trash Detection in Natural Environment Using a Deep Learning Model". In: *IEEE Access* (2023).

[9] M. Everingham et al. *The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results*. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[10] Zheng Ge et al. *YOLOX: Exceeding YOLO Series in 2021*. 2021. arXiv: 2107.08430 [cs.CV]. URL: https://arxiv.org/abs/2107.08430.

[11] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. *DropBlock: A regularization method for convolutional networks*. 2018. arXiv: 1810.12890 [cs.CV]. URL: https://arxiv.org/abs/1810.12890.

[12] Ross Girshick. *Fast R-CNN*. 2015. arXiv: 1504.08083 [cs.CV]. URL: https://arxiv.org/abs/1504.08083.

[13] Ross Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. arXiv: 1311.2524 [cs.CV]. URL: https://arxiv.org/abs/1311.2524.

[14] Gabriel Resende Gonçalves et al. "Benchmark for license plate character segmentation". In: *Journal of Electronic Imaging* 25.5 (Oct. 2016), p. 053034. ISSN: 1017-9909. DOI: 10.1117/1.jei.25.5.053034. URL: http://dx.doi.org/10.1117/1.JEI.25.5.053034.

[15] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. *Learning non-maximum suppression*. 2017. arXiv: 1705.02950 [cs.CV]. URL: https://arxiv.org/abs/1705.02950.

[16] Md Sazzad Hossain, John M. Betts, and Andrew P. Paplinski. "Dual Focal Loss to address class imbalance in semantic segmentation". In: *Neurocomputing* 462 (2021), pp. 69–87. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.07.055. URL: https://www.sciencedirect.com/science/article/pii/S0925231221011310.

[17] Gee-Sern Hsu, Jiun-Chang Chen, and Yu-Zu Chung. "Application-Oriented License Plate Recognition". In: *IEEE Transactions on Vehicular Technology* 62.2 (2013), pp. 552–561. DOI: 10.1109/TVT.2012.2226218.

[18] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. *Ultralytics YOLOv8*. Version 8.0.0. 2023. URL: https://github.com/ultralytics/ultralytics.

[19] Alexander Kirillov et al. *Segment Anything*. 2023. arXiv: 2304.02643 [cs.CV]. URL: https://arxiv.org/abs/2304.02643.

[20] Ivan Krasin et al. "OpenImages: A public dataset for large-scale multi-label and multi-class image classification." In: (2017).

[21] R. Laroca et al. "A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector". In: *International Joint Conference on Neural Networks (IJCNN)*. July 2018, pp. 1–10. DOI: 10.1109/IJCNN.2018.8489629.

[22] R. Laroca et al. "On the Cross-dataset Generalization in License Plate Recognition". In: *International Conference on Computer Vision Theory and Applications (VISAPP)*. Feb. 2022, pp. 166–178. ISBN: 978-989-758-555-5. DOI: 10.5220/0010846800003124.

[23] Chuyi Li et al. *YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications*. 2022. arXiv: 2209.02976 [cs.CV]. URL: https://arxiv.org/abs/2209.02976.

[24] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV]. URL: https://arxiv.org/abs/1405.0312.

[25] Sylwia Majchrowska et al. "Deep learning-based waste detection in natural and urban environments". In: *Waste Management* 138 (2022), pp. 274–284.

[26] Pedro F Proença and Pedro Simoes. "Taco: Trash annotations in context for litter detection". In: *arXiv preprint arXiv:2003.06975* (2020).

[27] Delong Qi et al. *YOLO5Face: Why Reinventing a Face Detector*. 2022. arXiv: 2105.12931 [cs.CV]. URL: https://arxiv.org/abs/2105.12931.

[28] Joseph Redmon and Ali Farhadi. *YOLO9000: Better, Faster, Stronger*. 2016. arXiv: 1612.08242 [cs.CV]. URL: https://arxiv.org/abs/1612.08242.

[29] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV]. URL: https://arxiv.org/abs/1804.02767.

[30] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV]. URL: https://arxiv.org/abs/1506.02640.

[31] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV]. URL: https://arxiv.org/abs/1506.01497.

[32] Roboflow. *License Plates Dataset*. URL: https://public.roboflow.com/object-detection/license-plates-us-eu.

[33] R. Schettini S. Bianco E. Gaviraghi. "Efficient Deep Learning Models for Litter Detection in the Wild". In: *8th IEEE International Forum on Research and Technologies for Society and Industry (RTSI)* (2024).

[34] Charlie Snell et al. *Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters*. 2024. arXiv: 2408.03314 [cs.LG]. URL: https://arxiv.org/abs/2408.03314.

[35] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG]. URL: https://arxiv.org/abs/1905.11946.

[36] Ultralytics. *YOLOv5: A state-of-the-art real-time object detection system*. https://docs.ultralytics.com. 2020. DOI: 10.5281/zenodo.3908559. URL: https://github.com/ultralytics/yolov5.

[37] Chengcheng Wang et al. *Gold-YOLO: Efficient Object Detector via Gather-and-Distribute Mechanism*. 2023. arXiv: 2309.11331 [cs.CV]. URL: https://arxiv.org/abs/2309.11331.

[38] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. 2022. arXiv: 2207.02696 [cs.CV]. URL: https://arxiv.org/abs/2207.02696.

[39] Jason Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. arXiv: 2201.11903 [cs.CL]. URL: https://arxiv.org/abs/2201.11903.

[40] Yuanjun Xiong et al. "Recognize Complex Events from Static Images by Fusing Deep Channels". In: *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE. 2015.

[41] Shuo Yang et al. *WIDER FACE: A Face Detection Benchmark*. 2015. arXiv: 1511.06523 [cs.CV]. URL: https://arxiv.org/abs/1511.06523.