

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



AI-Based, Real-Time Object Detection in the Public Landscape

André Vilhena da Costa

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Armando Jorge Sousa

Second Supervisor: Prof. Nuno Feixa Rodrigues

July 21, 2023

Resumo

Nas grandes cidades, as entidades responsáveis enfrentam a tarefa extremamente desafiadora de manter os espaços públicos limpos e seguros. Em muitos casos, estas entidades dependem dos seus próprios cidadãos para reportar problemas como a existência de resíduos na rua, de paredes vandalizadas com grafitti ou de sinais de trânsito danificados. No entanto, com a evolução de tecnologias como inteligência artificial e visão por computador, estas ocorrências podem ser detetadas automaticamente permitindo uma melhor deteção e acompanhamento dos possíveis problemas.

Este trabalho integra-se no projeto denominado Tidy City, cujo objetivo é desenvolver um sistema inteligente baseado em visão por computador, capaz de detectar e classificar automaticamente anomalias em espaços públicos. A arquitetura do projeto global consistirá num dispositivo móvel com uma câmara incorporada que pode ser instalado em qualquer veículo, onde um modelo de deteção de objetos estará a detetar pessoas e matrículas em tempo real. Estas anotações são censuradas antes de enviar as imagens para um servidor, no qual um segundo modelo vai analisar as imagens e detetar diferentes tipos de objetos - contentores de resíduos, sacos de resíduos, cartão, grafitti, sinais de trânsito e cartazes de publicidade; as imagens anotadas são posteriormente transferidas para um outro servidor, onde são armazenadas; e, por fim, as entidades responsáveis pela gestão e manutenção deste tipo de problemas vão ter acesso a uma aplicação *web* na qual vão ser apresentadas todas as deteções feitas na cidade - incluindo a respetiva classe do objeto e geolocalização - de modo a poderem atuar sobre o problema. Este trabalho foca-se no treino e teste dos dois modelos. O primeiro para a deteção de pessoas e matrículas em tempo real, de forma a permitir a privacidade "by design". E o segundo para a deteção das restantes classes de objetos consideradas relevantes para esta aplicação, sem o requisito da deteção em tempo real. O teste destes modelos foi realizado com dados recolhidos na cidade do Funchal, na Madeira.

Diferentes versões do modelo YOLOv7 foram treinadas para fazer a deteção de pessoas e matrículas em tempo real, recorrendo a datasets como o COCO e o Open Images. Após treinar com mais de 10000 imagens, os melhores resultados não foram obtidos com estes datasets, mas sim com cerca de 2200 imagens recolhidas na própria cidade do Funchal por parte de uma empresa de logística local. O YOLOv7-tiny em particular, revelou várias vantagens relativamente à velocidade de inferência e eficácia na deteção destes objetos, obtendo uma precisão média de aproximadamente 81%. Posteriormente, para a deteção das restantes classes de objetos foram utilizados alguns modelos de maior complexidade e, consequentemente, com uma maior eficácia, tal como o YOLOv7-e6e e o EfficientDet-D4. Utilizando *datasets* com mais de 5000 imagens, estes modelos foram capazes de atingir uma precisão média superior a 80% para todas as classes.

Palavras-chave: Visão por computador, deteção de objetos, deteção em tempo real

Abstract

In large cities, responsible entities face the extremely challenging task of keeping public spaces clean and safe. In many cases, these entities rely on their own citizens to report issues such as improper waste disposal, walls vandalized with graffiti, or damaged traffic signs. However, with the evolution of technologies like artificial intelligence and computer vision, these events can be automatically detected, allowing for better detection and monitoring of potential problems.

This work is included in the project called Tidy City, which goal is to develop a smart system based on computer vision capable of automatically detecting and classifying anomalies in public spaces. The overall project architecture will consist of a mobile device with an embedded camera that can be installed on any vehicle, where an object detection model will detect people and license plates in real-time. These annotations are blurred before the images are sent to a server, in which a second model will analyze the images and detect different types of objects - waste containers, waste bags, cardboard, graffiti, traffic signs, and advertising billboards. The annotated images will subsequently be transferred to a different server and stored. Finally, the entities responsible for managing and maintaining these types of problems will have access to a web application where all the detections made in the city will be presented, including the respective object class and geolocation, in order to take action on the problem. This work focuses on the training and testing of the two models. The first one performs real-time detection of people and license plates in order to ensure privacy "by design". And the second one detects the remaining classes of objects considered relevant for this application without the requirement of real-time detection. The testing of these models was carried out with data collected in the city of Funchal, in Madeira.

Different versions of the YOLOv7 model were trained to perform real-time detection of people and license plates using datasets such as COCO and Open Images. After training with over 10000 images, the best results were not obtained with these datasets but rather with around 2200 images collected in the city of Funchal by a local logistics company. YOLOv7-tiny, in particular, showed several advantages in terms of inference speed and accuracy in detecting these objects, achieving an average precision of approximately 81%. Subsequently, for the detection of the remaining classes of objects, some more complex and accurate models were used, such as YOLOv7-e6e and EfficientDet-D4. Using datasets with over 5000 images, these models were able to achieve an average precision of over 80% for all classes.

Keywords: Computer vision, object detection, real-time detection

Acknowledgements

First of all, I would like to thank my supervisor, prof. Armando Sousa, for giving me the opportunity to be part of such an interesting and forward-looking project. I would also like to thank my co-supervisor, prof. Nuno Rodrigues for being so supportive and patient with me every step of the way.

To all the friends that I was fortunate enough to make at FEUP throughout the last five years, I will be forever grateful for everything you taught me, and for every moment we shared with each other. We have grown so much together during some of the most difficult and challenging times of our lives so far. I am certain that I would not have been able to finish this master's degree with the same sense of fulfillment without your companionship.

I specially dedicate this work to my girlfriend, Bárbara, for everything we have been through together during such a challenging journey. Without all your support, patience, and understanding, I would never be who I am today. I have to apologize for every last-minute canceled date, but I honestly believe that we will look back and realize how everything was worth it.

Finally, all the hard work that I have devoted for the last five years and everything I was able to achieve for myself would not have been possible without the support of my family, in particular from my mother, my father, and my brother. You, more than anyone else, know that it was not easy to get here. The resilience and persistence I managed to develop were all thanks to the example you gave me every single day.

To all of you. Thank you.

André Costa

“I have been impressed with the urgency of doing. Knowing is not enough; we must apply. Being willing is not enough; we must do.”

Leonardo da Vinci

Contents

1	Introduction	1
1.1	Context	2
1.2	Motivation and Goals	2
1.3	Document’s Structure	2
2	Computer Vision-Based Object Detection	4
2.1	Performance Evaluation and Benchmark Datasets	6
2.2	ResNet	8
2.3	EfficientNet	9
2.4	EfficientDet	9
2.5	R-CNN	10
2.6	YOLO - You Only Look Once	12
3	Applications of Computer Vision-Based Object Detection	14
3.1	Datasets	14
3.2	Object Detection Kit (ODK)	16
3.3	Detect Waste in Pomerania	18
3.4	Mask R-CNN with TACO	21
3.5	Faster R-CNN	22
4	State-of-the-art Analysis	24
4.1	Datasets	24
4.2	Accuracy	25
4.3	Real-time Detection	27
5	Object Detection in the Public Landscape	28
5.1	System Architecture	28
5.2	Detection of People and License Plates	29
5.2.1	Datasets and Preprocessing	29
5.2.2	Models	31
5.2.3	Training Results	31
5.2.4	Summary	39
5.3	Detection of Waste, Graffiti, Traffic Signs, and Billboards	41
5.3.1	Datasets and Preprocessing	41
5.3.2	Models	42
5.3.3	Training Results	43
5.3.4	Summary	51

6	Discussion	52
6.1	Datasets	52
6.2	Accuracy	54
6.2.1	Detection of People and License Plates	54
6.2.2	Detection of Waste, Graffiti, Traffic Signs, and Billboards	58
6.3	Real-time Detection and Training Time	62
7	Conclusions and Future Work	64
7.1	Main Contributions	64
7.2	Final Remarks	65
7.3	Future Work	66
A	Evaluation Metrics	67
A.1	Evaluation Metrics for Binary Classification	67
A.2	Evaluation Metrics for Non-Binary Classification	68
A.3	Confusion Matrix	68
B	Object Detection Performance for Different Applications	69
B.1	Datasets' Representative Images	69
B.2	Pseudo-labelling Technique	70
B.3	Image Distribution per Dataset in Detect-waste+	70
B.4	Image Distribution per Class in Extended TACO	71
B.5	Image Distribution per Class in classify-waste+	71
B.6	Confusion Matrix for the EfficientNet-B2 Model	72
B.7	Bounding Box Sizes for TACO-10	72
B.8	Confusion Matrices for Mask-RCNN	73
C	Complementary Training Tools	74
C.1	K-Folds Cross-Validation Python Script	74
	References	75

List of Figures

2.1	Computer vision techniques [1]	4
2.2	(a) Shows the architecture of two-stage detectors. (b) Shows the architecture of one-stage detectors. [2]	5
2.3	Shortcut Connection in ResNet [3]	8
2.4	EfficientDet Architecture [4]	10
2.5	Flowchart of R-CNN [5]	10
2.6	Object Detection Example using YOLO [5]	12
3.1	Object Detection Kit system overview [6]	17
5.1	System Architecture	29
5.2	Example images from the datasets used for this task	30
5.3	Confusion matrix of training 4 in table 5.2	33
5.4	Testing images annotated using training 4 in table 5.2 (yellow bounding boxes for "license plate" and pink for "person")	34
5.5	Confusion matrix of training 5.1 in table 5.3	35
5.6	Testing images annotated using training 5.1 in table 5.3 (green bounding boxes for "license plate" and yellow for "person")	35
5.7	Confusion matrix of training 10.2 in table 5.4	37
5.8	Testing images annotated using training 10.2 in table 5.4 (pink bounding boxes for "license plate" and green for "person")	37
5.9	Confusion matrix of training 11.1 in table 5.5	39
5.10	Testing images annotated using training 11.1 in table 5.5 (brown bounding boxes for "license plate" and green for "person")	39
5.11	Example images from the datasets used for this task (in addition to the ones in figure 5.2)	42
5.12	Confusion matrix of training 5 in table 5.9	45
5.13	Testing images annotated using training 5 in table 5.9 (white bounding boxes for "plastic bag", green for "graffiti", brown for "waste container", grey for "traffic sign, and purple for "license plate")	46
5.14	Confusion matrix of training 7 in table 5.10	48
5.15	Testing images annotated using training 7 in table 5.10 (blue bounding boxes for "plastic bag", light green for "graffiti", pink for "cardboard", dark green for "waste container", white for "traffic sign, and orange for "license plate")	49
5.16	Confusion matrix of training 12.1 in table 5.11	50
5.17	Testing images annotated using training 12.1 in table 5.11 (blue bounding boxes for "plastic bag", purple for "cardboard", red for "waste container", yellow for "person", green for "traffic sign, and blue for "license plate")	51

6.1	A compilation of testing images obtained with three different YOLOv7 models .	56
6.2	Comparison between the testing images achieved in training 4 and 11.1 (in tables 5.2 and 5.5)	57
6.3	Comparison between the testing images achieved in training 4 and 5 (in table 5.9)	59
A.1	Visual representation of the precision and recall metrics for binary object classification (Adapted from [7])	67
A.2	Visual representation of the precision, recall and IoU metrics for non-binary object classification [8]	68
A.3	An example of a confusion matrix with three different classes of flowers [9] . . .	68
B.1	Representative images from each dataset [10]	69
B.2	Flowchart for pseudo-labeling semi-supervised learning technique [10]	70
B.3	Image distribution per class in classify-waste+ [10]	70
B.4	Image distribution per class in Extended TACO [10]	71
B.5	Image distribution per class in classify-waste+ [10]	71
B.6	Confusion matrix for EfficientNet-B2 with weighted sampler and pseudo-labelling per batch [10]	72
B.7	Histogram of bounding box sizes per category for TACO-10 [11]	72
B.8	Confusion matrices for matches with $\text{IoU} > 0.5$. (Left) detections with $\text{ratio_score} > 10$, (Right) detections with $\text{ratio_score} > 50$. BG represents background. [11]	73

List of Tables

3.1	Accuracy results of the object detection experiment (Adapted from [6]). P: precision; R: recall; F1: f1-score.	17
3.2	Accuracy results of selected architectures on detect-waste with one class - litter [10]	19
3.3	Results on different datasets using EfficientDet-D2 [10]	19
3.4	EfficientDet-D2 final evaluation results [10]	20
3.5	Accuracy results of selected classifiers (Adapted from [10])	20
3.6	Summary of the precision, recall and F1-score for each waste class [10]	21
3.7	AP for 4-fold cross validation using different scores (%) to rank predictions [11] .	22
3.8	Number of images and instances per class [12]	22
3.9	Classification performance for Faster R-CNN models (Adapted from [12])	23
3.10	Accuracy results for each class using Faster R-CNN models (Adapted from [12])	23
5.1	State-of-the-art accuracy results using YOLOv7 models, by ascending order of complexity	31
5.2	Accuracy results using the YOLOv7-tiny model	32
5.3	Accuracy results using the YOLOv7-e6 model	34
5.4	Accuracy results using the YOLOv7-e6e model	36
5.5	Accuracy results using the YOLOv7-tiny model with an updated dataset	38
5.6	Accuracy results for the best-performing models using annotated batches 1, 2, and 3 as training dataset - first task	40
5.7	Inference time for the best-performing models of each YOLOv7 version	40
5.8	State-of-the-art accuracy results using YOLOv7 and EfficientDet models	43
5.9	Accuracy results using the YOLOv7-e6e model to detect 7 classes	44
5.10	Accuracy results using the YOLOv7-e6e model to detect 8 classes	47
5.11	Accuracy results using EfficientDet models to detect eight classes	50
5.12	Accuracy results for the best-performing models - second task	51

Chapter 1

Introduction

Maintaining public spaces and infrastructures has become increasingly challenging as the number of people living in cities outgrows the amount of available resources. Inevitably this phenomenon leads to poor maintenance, which directly affects public safety and health. Hence, local governments and private companies in charge of cities' maintenance are faced with the extremely complex task of managing these resources as efficiently as possible.

The most common public issues encountered in large cities are improper waste disposal and damaged infrastructures such as walls and traffic signs - either from vandalism, over-use, or natural disasters. Improper waste disposal, besides being a potential threat to public health, also affects the visual aspect of a city, which is a major concern, particularly for local governments with an interest in investing in the city's tourism. Moreover, graffiti detection could be used to schedule clean-ups automatically. On the other hand, damaged traffic signs might represent a serious threat to the drivers' safety and, consequently, to public safety. In order to tackle these problems, a significant investment would have to be made in public surveillance systems and in the maintenance personnel - responsible for fixing most of these issues. At present, there are two potential solutions for implementing such a surveillance system. The first involves deploying field staff to monitor and report on these occurrences, acting as on-ground observers. The second requires the establishment of an extensive city-wide video surveillance infrastructure, which, in turn, would need a team of human operators to interpret the data. However, both these solutions are highly inefficient due to their dependency on human employees, also resulting in very limited system scalability.

Considering the above-mentioned limitations of current solutions, technology might give an important contribution to improving efficiency and reducing costs, namely by taking advantage of computer vision-based systems. The purpose of this project is to develop a system capable of automatically detecting and classifying multiple classes of objects and certain issues in public spaces. The main advantage of this approach is the smaller investment required since existing cameras can be used, and a much simpler surveillance system can be implemented using smartphones, for example. New datasets will also be built considering Madeira's landscape in order to train our models. In the end, the model will be tested on the field and tuned accordingly until it reaches the best performance possible.

1.1 Context

The present work is being developed with INESC TEC (Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência), having prof. Nuno Feixa Rodrigues as the project's coordinator.

INESC TEC is a private non-profit research association known for its dedication to scientific research and technological development in the domains of Computer Science, Industrial, and Systems Engineering, among others. Over its 35 years of existence, it has been the birthplace of many innovative projects that tackle "real-world" problems. Combining this with the rising popularity of computer vision applications and their developments, there are good foundations to invest in such a project.

1.2 Motivation and Goals

Keeping the cities clean and safe is one of the main goals of this project. However, with the constant rising of cities' population, responsible entities are challenged to adapt their services to be as efficient as possible and consequently cheaper. The dependency on human input is the main reason for this lack of efficiency. Therefore, investing in computer vision technology capable of detecting a wide range of issues might be a good solution for these entities.

Recent developments regarding this type of technology have made it much faster and more portable. Nevertheless, there is still very little research on its applications in the public landscape. The main challenges that researchers might be faced with are the different shapes and sizes that the same object can have, along with the very particular backgrounds found inside a city. Hence, this project may not only provide some answers to these problems but also a platform that motivates further research.

1.3 Document's Structure

The remaining of this document contains six more chapters. Their content is summarized below.

Chapter 2 The state of the art in object detection techniques and deep-learning models is explored in this chapter. It starts by explaining some evaluation metrics, as well as introducing cross-validation techniques and various benchmark datasets. Relevant theoretical background is also laid out in order to facilitate the comprehension of the following chapters. In the end, there is a summary of the general results that each model achieved during its development.

Chapter 3 Some related works in the field of object detection in public spaces, mainly focused on waste detection and classification, are thoroughly explained. Their corresponding results are displayed and discussed individually.

Chapter 4 Considering the results of chapter 3, a comparison between each approach is made. Some of the topics taken into consideration are the model's accuracy and inference speed. The

main goal of this discussion is to evaluate which model (or models) is expected to fit the use case of this work better and why. This chapter includes the most relevant results mentioned in chapter 3, so its reading can be done independently from the previous chapter.

Chapter 5 In this chapter, the system's architecture chosen for this project is displayed, and every training iteration developed is thoroughly described. Every dataset and model used for each training is laid out, as well as the corresponding results achieved.

Chapter 6 Considering the results mentioned in chapter 5, a comparison between each model is made considering the datasets used, the detection accuracy achieved, and the corresponding inference and training time. The main goal of this discussion is to evaluate which model (or models) is expected to fit each task better and why. This chapter includes the most relevant results mentioned in chapter 5, so its reading can be done independently from the previous chapter.

Chapter 7 The conclusions drawn from all the experiments and corresponding results mentioned in previous chapters are highlighted here. A critical analysis of the achievements observed during this project's development and some topics that can be further improved in future iterations are also laid out.

Chapter 2

Computer Vision-Based Object Detection

"Object detection is one of the most important and challenging branches of computer vision" [2], which has been attracting increasing amounts of attention in recent years due to its wide range of applications. On top of image classification - when the class of objects present in an image is predicted - object localization is responsible for locating these objects with a bounding box and displaying their corresponding classes, such as human faces, traffic signs, or vehicles. Apart from these two methods, image segmentation can also be applied - when a pixel-wise mask for each object is created to give more information about its shape.

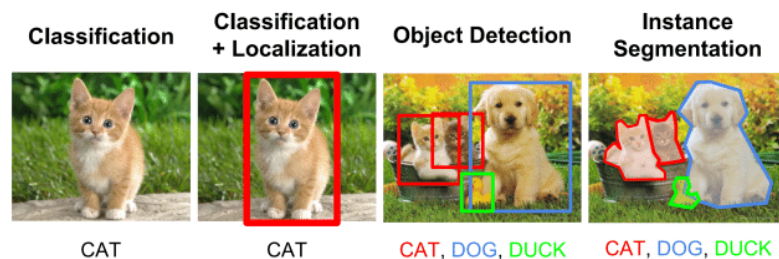


Figure 2.1: Computer vision techniques [1]

As knowledge regarding computer vision algorithms keeps growing, deep learning models in particular have been widely adopted for object detection applications. Deep learning is a part of machine learning, represented by neural networks with three or more layers. These neural networks attempt to simulate the behavior of the human brain, allowing it to learn from large amounts of data and make more accurate predictions than single-layer networks [13]. After being trained with suitable datasets, these models are responsible for locating and classifying the existing objects in an image as quickly and efficiently as possible. To do so, they rely on object detectors.

Object detectors can be divided into two categories: one-stage and two-stage detectors (2.2). YOLO and Faster R-CNN are the most representative models of each one, respectively [2]. One-

stage detectors start by predicting regions of interest where it is likely to find an object. Then, features from each candidate region are extracted by a RoIPool (Region of Interest Pooling) layer and are finally classified as one of the known classes that were defined for the model in question, for instance, "person", "trash", or "car". On the other hand, the two-stage detectors have a second stage called RPN (Region Proposal Network), which is applied before the RoIPool layer and which is responsible for proposing possible regions of interest to be selected and further processed. This is what makes one-stage detectors more time efficient and, consequently, more suitable for real-time applications than the ones with two stages.

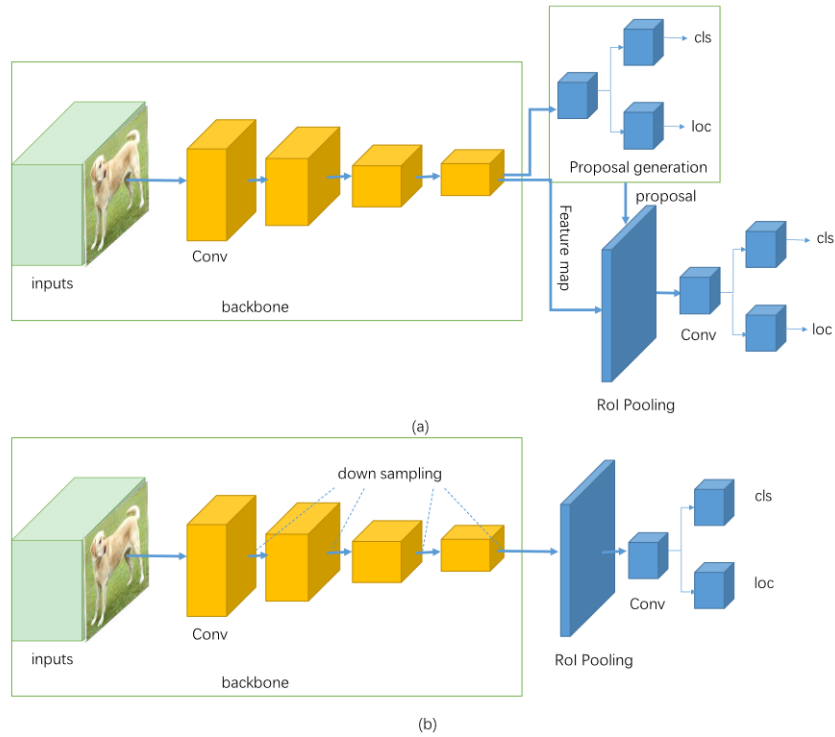


Figure 2.2: (a) Shows the architecture of two-stage detectors. (b) Shows the architecture of one-stage detectors. [2]

On top of object localization, the architecture of these models can also include a classification layer where a specific class is assigned to each of the objects present in the input image. This layer usually relies on algorithms such as SVM (Support Vector Machine); or on different types of CNNs, such as ResNet and EfficientNet. During the literature review, the most recent works found regarding object detection were all based on the last two. Therefore, only these are going to be further discussed in this chapter.

2.1 Performance Evaluation and Benchmark Datasets

In the following sections of this chapter, a brief theoretical background of some state-of-the-art computer vision models will be explained. In the end, the most relevant accuracy results for each one of these models are also laid out. The evaluation metrics and techniques used in each reference may vary since they depend on the datasets in question. Therefore, this section will be used to share some fundamental knowledge about these topics, as well as some benchmark datasets available for object detection.

Metrics rely on the interpretation of labels as true or false positives and true or false negatives. For binary classification, **precision** represents the ratio of true positives over all the predicted labels, while **recall** defines the relation between the number of true positives and false negatives together with true positives (figure A.1). On the other hand, for non-binary classification, **precision** represents the ratio of the intersection between the predicted box and the ground truth box over the detected box's total area. The averaged value of precision for each class results in another commonly used metric called **mAP** (mean average precision). **Recall** then defines the relation between this intersection and the ground truth box's area (figure A.2). In addition, the **f1-score** is the harmonic mean of both these values, which describes the relationship between them.

In non-binary classification, another metric is used to assess the accuracy of the predicted bounding boxes, named **Intersection over Union or IoU**. It represents the ratio of intersection between the ground truth and the predicted box over the union of both boxes' areas (figure A.2). The higher the value (between 0 and 1), the better the prediction. This metric is used as a threshold from which a prediction is considered true, e.g. mAP@0.50 means that a prediction corresponds to a true positive label if the IoU is at least 0.50 (this is the standard threshold used, so it can also be simply called mAP; or AP, for binary classification). Other commonly used representations are mAP@0.75 and mAP@0.50:0.95. The last one considers an IoU threshold between 0.50 and 0.95 with a step of 0.05, which means it calculates ten different AP values that are then averaged in order to give a single value. There are also metrics that consider the object's size, such as AP_S (small), AP_M (medium), and AP_L (large).

Before being able to analyze each of the previous evaluation metrics, a machine learning model needs to be cross-validated in order to know if it is underfitting or overfitting the data. In other words, **cross-validation** gives a reference of how well the model will generalize to an unseen dataset. There are several different techniques, but all of them follow a similar algorithm: splitting the dataset into training and testing datasets (the way this split is done depends on the technique used), training the model on the training dataset, and validating the model on the validation dataset. The most commonly used and most simple cross-validation technique is the hold-out, which randomly splits the dataset regardless of each class' representativity. However, this might not be the best method to use on datasets with very unbalanced classes. To prevent splitting the dataset in a way that favors a specific class during training (probably the one with the most amount of annotations), k-folds cross-validation appears to be the most suitable technique for this task.

K-folds cross-validation is based on the following algorithm [14]:

1. Pick a number of folds ("k"). It must be smaller than the dataset's length;
2. Split the dataset into k equal (if possible) parts/folds;
3. Choose k-1 folds as the training set. The remaining fold will be the validation set;
4. Train the model on the training dataset;
5. Validate on the validation dataset;
6. Save the validation results;
7. Repeat steps 3 to 6 "k" times. Each time the remaining fold is used as the testing dataset. In the end, the model should have been validated on every fold;
8. Average the results achieved in step 6 to get the final results.

After cross-validating a model, there is one last tool that can be used to evaluate its classification performance - the confusion matrix. This matrix displays the number of true positives, true negatives, false positives, and false negatives produced by the model on a test dataset. For binary classification, the matrix will be a 2x2 table, while for multi-class classification, the matrix shape will be a square with the same number of rows as the number of classes detected. Yet, the values displayed on confusion matrices are very dependent on the confidence threshold (the minimum confidence any prediction has to achieve in order to be considered a true positive prediction) selected, which makes it less reliable for comparing results between different object detection models. An example of a standard confusion matrix is presented in figure A.3.

In object detection, there are three main datasets to consider for performance evaluation: ImageNet [15], PASCAL VOC [16], and MS COCO [17]. A brief description of each one of them follows.

ImageNet. ImageNet [15] contains 1000 object classes in over 1.2 million images. In total, there are more than 500000 object instances. The classes are very detailed, including for example, several types of vehicles or even dog breeds.

Pascal VOC. This dataset contains 20 object categories spread over 11000 images, with more than 27000 annotations [16]. These categories are split into four main branches - vehicles, animals, household objects, and people. The last release occurred in 2012. When training an object detection model with this dataset, the average precision is only calculated using the standard IoU threshold, 0.50.

Microsoft COCO. COCO (Common Objects in Context) [17] is a benchmark dataset widely used to evaluate the performance of an object detection model when considering many different

classes of objects in different types of natural backgrounds. The last release (2017) includes more than 150000 images with over 80 object categories. When using COCO, metrics such as $\text{mAP}@0.50$, $\text{mAP}@0.75$, and $\text{mAP}@0.50:0.95$ are used, as well as the previously mentioned AP_S , AP_M and AP_L [18].

2.2 ResNet

One of the main issues in designing CNNs is scaling the model in order to increase its accuracy. One might be tempted to believe that the deeper a neural network is, the better it will perform. However, when the network's depth keeps increasing, its accuracy reaches a saturation point and then starts degrading rapidly [3]. Such degradation might be confused with overfitting, but in fact, adding more layers to an already deep network eventually leads to higher training errors. This is caused by the notorious vanishing gradients problem, according to which, when a network becomes too deep, the gradients used to calculate the loss function approach zero, leading to worse training performance.

To tackle this problem, ResNets use shortcut connections that connect specific layers to further layers while skipping some in between. These residual connections don't pass through as many activation functions, and therefore the gradients are not as affected as previously, resulting in less performance error. In figure 2.3, there is an example of a shortcut connection.

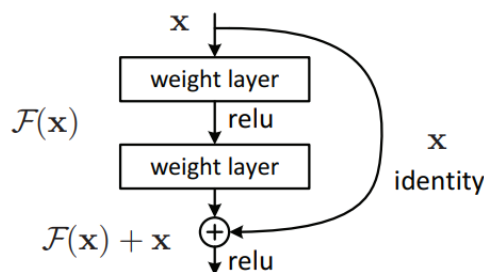


Figure 2.3: Shortcut Connection in ResNet [3]

In [3], some experiments were carried out in order to evaluate the classification performance of these networks. For instance, they trained it on the ImageNet 2012 classification dataset. Around 50000 validation images were used to evaluate the model considering both top-1 and top-5 error rates. The best performances were achieved by ResNet-152 and ResNet-101, with a top-1 error of 19.38% and 19.87% and a top-5 error of 4.49% and 4.60%, respectively. These results confirm the advantages of adding depth to a model.

2.3 EfficientNet

The most common way to scale up a CNN is by its depth, width, or even by the input's resolution. While ResNets only rely on the number of layers to scale a model (depth), the authors of [19] have come to the conclusion that in order to perform model scaling efficiently, all of these three parameters must be considered. Their method - compound scaling - uniformly scales each of these dimensions with a set of fixed coefficients in the following way:

$$\begin{cases} depth = \alpha^\phi \\ width = \beta^\phi \\ resolution = \gamma^\phi \end{cases} \quad (2.1)$$

considering $\alpha \geq 1, \beta \geq 1, \gamma \geq 1$, such that $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$

The ϕ coefficient is specified by the user depending on how many more resources are available to train the model. On the other hand, α , β , and γ specify how these extra resources are distributed by each of the network's dimensions - depth, width, and resolution, respectively.

EfficientNets' state-of-the-art performance for image classification is well represented in [19], where several versions of this model were trained on the ImageNet dataset. As an example, the biggest (or most up-scaled) model - EfficientNet-B7 - achieved around 84.3% top-1 and 97.0% top-5 accuracy. On top of that, this model is 8.4 times smaller and 6.1 times faster than the previous best CNN model.

2.4 EfficientDet

After the success EfficientNets demonstrated for object classification, a similar approach was developed for object detection, known as EfficientDets. Similar to an EfficientNet, an EfficientDet tries to answer the question of how to "build a scalable detection architecture with both higher accuracy and better efficiency across a wide spectrum of resource constraints" [4].

EfficientDet's architecture mainly follows the one-stage paradigm previously mentioned. It is divided into three different parts - the backbone network, the feature network, and the class and box network (see figure 2.4). On top of that, it follows the steps of EfficientNet by compound scaling each of these networks to achieve better accuracy with fewer parameters.

The first part relies on EfficientNets pre-trained on the ImageNet dataset. To extract the input images' features, a new approach is proposed in [4], known as BiFPN (Weighted Bi-directional Feature Pyramid Network). BiFPN is a type of FPN that allows easy and fast multi-scale feature fusion by adding a specific weight for each input feature, allowing the network to learn the importance of each one; lastly, these fused features are fed to the class and box networks that generate the object's class and bounding box predictions, respectively.

To evaluate this model, the authors of [4] trained it on COCO 2017 detection datasets (with 118000 training images [18]). Considering a low-accuracy performance model, EfficientDet-D0

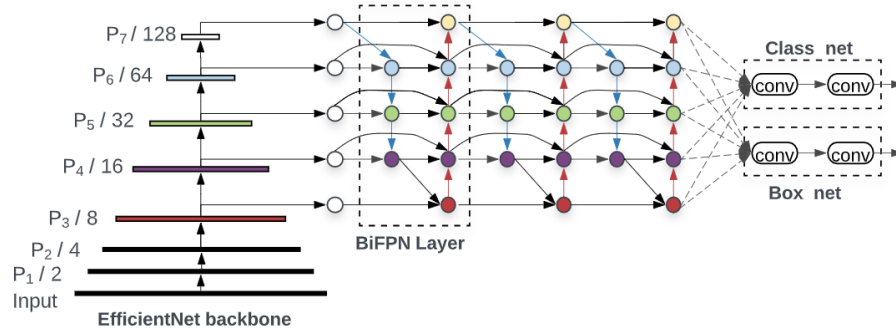


Figure 2.4: EfficientDet Architecture [4]

achieved similar accuracy as YOLOv3 (53.0%) with 28 times fewer FLOPs (Floating Point Operations) - the higher the value, the slower the model. On the other hand, the EfficientDet-D7x model achieved state-of-the-art accuracy of 74.3% when tested on COCO's test dataset (with 41K testing images).

2.5 R-CNN

The first two-stage detector to show that a CNN was able to achieve great performances regarding object detection was the R-CNN, a region-based CNN detector [2]. This detector consists of four modules. The first generates about 2000 region proposals for each image by relying on selective search [5]. The second module extracts a fixed-size feature vector from each one of these proposals. The third is a set of linear SVMs that classify the objects. And the last module is a bounding-box regressor that is responsible for precisely highlighting the object on the image with a box around it.

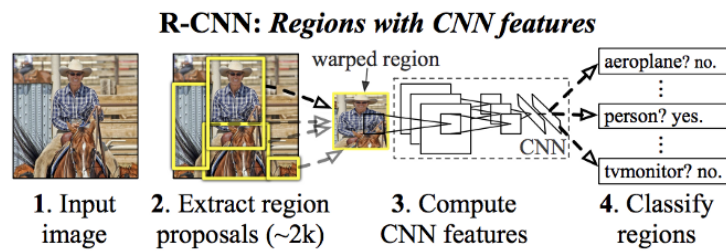


Figure 2.5: Flowchart of R-CNN [5]

A year after R-CNN was proposed, a faster version was developed, named Fast R-CNN. While R-CNN inputs each region proposal to the CNN, the newer version is able to extract all the features from an image at once and send them to the CNN for classification, which saves a large amount of time on processing (it is 9 times faster [2]). And a few months later, Faster R-CNN was proposed

to replace the selective search - used for proposing RoI - with an RPN, a convolutional network capable of efficiently predicting region proposals.

In addition to the last two versions, the R-CNN family also has a model developed for object segmentation, named Mask R-CNN. This one still follows the two-stage architecture, with an RPN as the first stage. However, in the second stage, it outputs a binary mask for each RoI, as well as the corresponding class and bounding box. This is achieved by adding another layer called RoIAlign, which removes the misalignments between the RoI and the extracted features previously introduced by the RoIPool layer. Even though these misalignments don't have a big impact on classification, they do have on predicting pixel-accurate masks (needed to perform segmentation) [20].

The authors of [5] did a very extensive experimental evaluation on several different models, including Faster R-CNN. One of these experiments was to test the classification accuracy of this model on the PASCAL VOC 2007 (VOC2007) test dataset, which contains around 5000 images and 12000 objects in total [21]. The model was trained in three different ways. Firstly, it was trained on the VOC2007 training/validation dataset (with around 5000 images and 13000 objects [21]); then it was trained with a union of VOC2007 and VOC2012 training/validation datasets (around 17000 images and 40000 objects combined [16]); finally, it was trained on COCO 2014 training/validation dataset (with around 120000 images [17]) and then fine-tuned on VOC2007 and VOC2012. Considering the 20 different object categories the VOC2007 test dataset contains, the overall mAP for each of these tests was 69.9%, 73.2%, and 78.8%, respectively. In addition to these tests, they also trained two different configurations of the Faster R-CNN model on VOC2007 and VOC2012 in order to compare their testing speed on the VOC2007 test dataset. The first one, using the standard backbone network (a VGG16), achieved a mAP of 73.2% at a test rate of 9.1 frames per second. On the other hand, the second one used a ResNet101 and, despite achieving a better accuracy (83.8%), the test rate decreased significantly to 0.4 FPS.

Another experiment was made using the COCO test dataset, composed of more than 40000 images with a total of 80 object classes [17]. The objects from this dataset have a broader range of scales, which makes it more challenging than either VOC2007 or VOC2012. The target of this experiment is object localization accuracy, so the performance of each model is evaluated by AP computed under different degrees of IoU and on different object sizes. In this case, the Faster R-CNN model was only trained on the COCO training/validation dataset, and it performed poorly, only achieving a mAP@0.5 of 45.3%, which is equivalent to PASCAL VOC's metric - mAP. This experiment also evaluated the segmentation accuracy of some Mask R-CNN models with different frameworks. The best accuracy was 62.3% (considering 50% of IoU), achieved by the one that included a ResNeXt101 and an FPN in its architecture.

2.6 YOLO - You Only Look Once

As previously mentioned, even though two-stage detectors have increased time efficiency considerably in recent years, the architecture of one-stage detectors makes them the most suitable for real-time object detection applications. YOLO (You Only Look Once) is the main example in this field, since it was the first model to accomplish real-time detection on images and videos [2].

The model works by first splitting the input image into a grid of cells, where each cell is responsible for predicting a bounding box if its center falls within the corresponding cell. Each grid cell predicts X bounding boxes (with their corresponding coordinates, width, and height), as well as the confidence and class probability for each of them (see figure 2.6). For this reason, YOLO predicts less than 100 regions of interest - instead of 2000, like the Fast R-CNN - which allows running the model at 45 frames per second [2], something never seen until then. However, this model is not perfect since it doesn't perform very well for object localization within the image and multiple-object detection (when these are in a single grid cell), which led to the development of some improved versions.

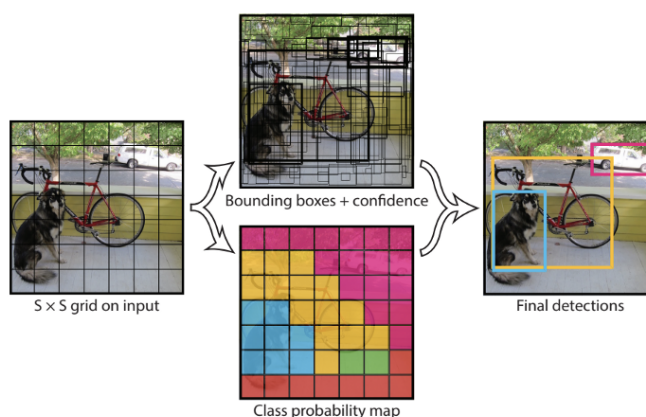


Figure 2.6: Object Detection Example using YOLO [5]

YOLOv2 brought the first improvement by enabling the detection of objects from over 9000 different categories, even though it was still not able to accurately detect smaller-sized objects [22]. Later, YOLOv3 offered a better detection performance considering smaller objects as well as overlapping objects [23]. Many other versions followed, each focused on increasing the processing speed as much as possible without affecting the model's performance accuracy. YOLOv7 was the most recently published version (in July of 2022 [24]) by the time this work was started, capable of surpassing all known object detectors in both speed and accuracy in the range from 5 FPS to 160 FPS. However, during this work's development, a more recent version was published - YOLOv8 [25].

Despite its detection speed, YOLO-based models generally don't achieve an accuracy as high as two-stage detectors. In [26], the YOLO model is trained on VOC2007 and VOC2012 training/validation datasets and then tested on VOC2007 test dataset, only achieving a mAP of 63.4%

(while running at 45 FPS). A YOLO model with VGG16 as its backbone was also trained in the same conditions, resulting in a better accuracy (66.4%) but also on a slower model (21 FPS), falling slightly below the real-time threshold. On the other hand, YOLOv3 [23] was trained and tested on the COCO dataset, reaching a mAP@0.50 of 57.9% at around 20 FPS (on a Pascal Titan X [27]). Furthermore, YOLOv5 [28] achieved 72.7% on the COCO validation dataset at 50 FPS (on an NVIDIA V100 GPU). In addition, according to [28], the latest version of YOLOv5 segmentation models are also the "fastest and most accurate", reporting a mAP@0.50:0.95 of 41.4% at over 200 FPS (on an NVIDIA A100 GPU). Finally, the best version of YOLOv7 achieved a mAP@0.5 of 74.4% at 36 FPS (on an NVIDIA V100 GPU).

Chapter 3

Applications of Computer Vision-Based Object Detection

There is still a very limited amount of research regarding object detection in an urban context, especially waste detection. As a consequence, there is also a lack of image datasets (with the corresponding annotations), that are required to train the models. After a literature review, the most relevant datasets that are publicly available, considering the problem in question, are laid out in section 3.1.

3.1 Datasets

TACO. TACO (Trash Annotations in Context) [11] is a crowd-sourced dataset of waste "in the wild" images mostly taken with high-resolution mobile phone cameras. At the time of writing, it contains 1500 annotated images with almost 5000 segmentation instances, including the corresponding bounding boxes. The annotations are provided in the COCO format [17] with an additional description of the background - "vegetation", "sand", "water", etc. These are divided into 60 categories of litter, which belong to 28 supercategories such as "bottle", "paper" and "plastic container". All objects can also be classified as only one class - litter.

Detect-waste+ Detect-waste+ is a dataset developed for the "Detect Waste in Pomerania" project [10], whose goal was to accurately detect waste on the streets of Gdansk (Poland). It combines eight publicly available datasets of waste in many different contexts. This compilation includes the following datasets: Extended TACO [11], UAVWaste [29], TrashCan [30], Trash-ICRA [31], MJU-Waste [32], Drinking Waste [33], Wade-AI [34] and Cigarette Butt [35]. For a better understanding of section 3.3, a brief description of each of these datasets will follow.

Extended TACO. Apart from the 1500 annotated images that TACO provides, it also offers around 3000 unannotated images. The authors of [10] took advantage of that and provided annotations only on the detection level, resulting in this new dataset with over 14000

instances in total (see figure B.4).

UAVVaste. UAVVaste [29] contains 772 manually-labeled aerial images of waste with over 3700 object instances of only one class - "rubbish". It also provides instance segmentation masks in the COCO format. The data was collected in streets, parks, and lawns using UAVs (Unmanned Aerial Vehicles).

TrashCan and Trash-ICRA. TrashCan [30] and Trash-ICRA [31] are datasets of underwater waste. Both contain frames of videos taken by ROVs (remotely operated underwater vehicles). The TrashCan dataset is annotated on the instance segmentation level and is composed of 7212 images with 16 classes, while Trash-ICRA is annotated on the detection level and contains 7668 images with 7 classes.

MJU-Waste. This dataset [32] is composed of 2475 indoor waste images manually labeled with instance masks in COCO format. There are two classes - waste and background. Objects are hand-held and mostly centered with the image. In most cases, there is only one object per image.

Drinking Waste. It contains more than 4800 images of drinking waste in an indoor and homogeneous background, taken with a mobile phone camera. The data belong to four classes - "aluminum cans", "glass bottles", "PET bottles" and "HDPE". The annotations are provided as bounding boxes in YOLO format.

Wade-AI. Wade-AI [34] contains almost 1400 images of waste "in the wild", obtained with Google Street View. There are 2200 manually-labeled instance masks [10] annotated in COCO format, with only one class - "rubbish". By looking at the images that compose the dataset, they were mainly taken from random places using Google Street View.

Cigarette butts. It consists of more than 2000 images where cigarette butts were artificially placed on different background images [35]. It provides annotations on the segmentation level.

Classify-Waste+. Classify-waste+ was also developed by the authors of [10] and it is also a compilation of different datasets: TrashNet [36], Open Litter Map [37], Waste Pictures [38] and Places [39]. While detect-waste+ was built with the purpose of object localization, this dataset was built for object classification. Once again, a brief description of each dataset will follow.

TrashNet. TrashNet [36] is composed of more than 2100 indoor pictures taken by a mobile phone. In most cases, a white background was used. The objects are divided into six categories - "glass", "paper", "cardboard", "plastic", "metal" and "trash".

Open Litter Map. Open Litter Map [37] is a free, open, and crowd-sourced dataset with more than 100000 images from all over the world, also taken by mobile phones. Each picture

provides information about the type (or types) of litter present in it, the exact coordinates where it was taken, the timestamp, and the phone model.

Waste Pictures. It contains around 24000 images [38], divided into 34 classes that were scraped from Google. This dataset is very diverse, including very unusual objects such as X-rays and drawings of waste.

Places. This is a very diverse compilation of 10 million scene photographs [39] taken from online search engines such as Google Images, Bing Images, and Flickr. In this case, the main goal of this dataset is to improve the distinction between waste and background.

3.2 Object Detection Kit (ODK)

Urban issues are very frequent and can't be tracked if street-level images are only recorded at long time intervals, which makes Google Street View and similar platforms unreliable for this task. Therefore, the Object Detection Kit framework was developed in order to collect images at a very high frequency.

The authors of [6] proposed a system that works by using an "app" running on smartphones attached to local governments' vehicles, such as waste trucks and police cars, while they drive around the city doing their normal business. It can also be expanded to volunteers' vehicles or even integrated into cars that are already equipped with cameras, which is getting increasingly common due to the rapid evolution of autonomous driving cars. As with any other machine learning system, this one also needs very large amounts of data in order to be accurate, which implicates a large number of streaming clients (the ones capable of collecting data), as well as a large amount of manually annotated images. Currently, the most feasible way to achieve the first aspect is by relying on the low cost and ubiquity of smartphones, and therefore using their cameras as the data collectors. On the other hand, the efficiency of the annotating and training process also requires further improvement in order to allow scalability.

The system is divided into 3 main parts, namely the streaming clients; the distribution server, where the frames are analyzed; and a database where the processed images are stored and then shared with a dashboard, as seen in figure 3.1.

After setting up the "app" on a smartphone, the streaming client is ready to start scanning the surrounding environment, either automatically (by detecting the vehicle moving) or manually; whenever it starts, the vehicle's license plate is registered, in order to identify the streaming client; then 3 frames per second are sent constantly to the distribution server, combined with the corresponding geographical location and timestamp - this server is the core of the whole system since it is responsible for the communication between a dynamic number of streaming clients and the frame analyzers inside the server; finally, all the frames stored in the distribution server are analyzed by several machines called frame analyzers, where a machine learning model starts by filtering sensitive information, such as human faces and license plates, and then it detects the objects in each image. Furthermore, the output data from the frame analyzers is used to create a

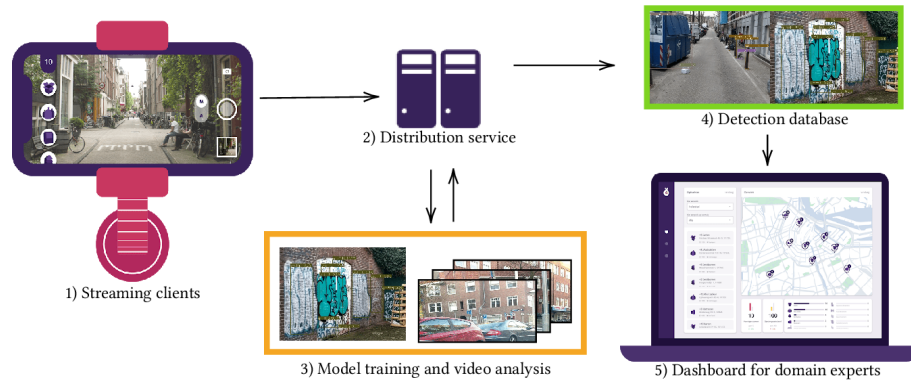


Figure 3.1: Object Detection Kit system overview [6]

geographical interface, where the local authorities assigned to this task can see when and where each class of objects was found throughout the day. Hence, they are able to monitor how clean the streets are, understand which are the most efficient cleaning routes, and even detect which city areas are needing more intervention. The first developed version of the Object Detection Kit relied on a YOLOv3 model [6] to build the system previously mentioned. An improved version that uses YOLOv5 [28] was already published since then. However, there is no paper or article reporting the achieved results yet, only the code is publicly available [40].

As a performance reference for ODK, the YOLOv3 model was trained on a small dataset of 1400 labeled images taken on the streets of Amsterdam [6], 1288 of which were used to train the model, while the remaining 112 images were used as the test dataset. In table 3.1, an overview of the classes present on the training and testing dataset can be found, as well as the corresponding accuracy results. Overall, the model has a relatively high accuracy of 61% for all classes.

Table 3.1: Accuracy results of the object detection experiment (Adapted from [6]). P: precision; R: recall; F1: f1-score.

Class	N. of training objects	N. of testing objects	P	R	mAP	F1
All	7986	514	0.21	0.74	0.61	0.31
Container small	1251	120	0.38	0.95	0.94	0.54
Garbage bag	1154	92	0.13	0.87	0.70	0.23
License plate filter	958	86	0.25	0.78	0.60	0.38
Face filter	894	62	0.22	0.79	0.62	0.34
Cardboard	888	59	0.18	0.59	0.43	0.28
Pole	751	43	0.12	0.86	0.73	0.21
Graffiti	706	39	0.47	0.95	0.87	0.63
Christmas tree	180	12	0.11	0.83	0.59	0.19
Mattress	50	1	0	0	0	0

3.3 Detect Waste in Pomerania

"Detect Waste in Pomerania" [10] was a project developed in Poland that tried to identify waste on the streets of Gdansk, while offering a solution to the lack of data available for waste detection "in the wild". The detection process was divided into two steps - litter localization and litter classification. The classification was based on a waste segregation system in Gdansk, which separates the waste into seven different categories - "bio", "glass", "metal and plastic", "non-recyclable", "other" (e.g. batteries, large household appliances, tires, etc.), "paper" and "unknown" (decomposed waste). During the development of this project, the authors combined twelve publicly available datasets with additional data collected using the Google Images Download tool [41]. For the first step, the segmentation and detection datasets [11, 29, 30, 31, 32, 33, 34] were combined resulting in the new detect-waste dataset, with a single class - "litter". Later, with the addition of the cigarette butt dataset [35] to boost the detection performance of small objects, it was renamed detect-waste+. For the classification stage, they combined litter instances cropped from some images used in the previous stage [11, 33] and added some other images of "bio", "glass", "other" and "paper" from the datasets [36, 38], in order to balance the representativity of each class - forming a classification dataset named classify-waste. A classify-waste+ dataset was also created by adding more than a thousand images containing waste-free backgrounds [39] and around 55000 pseudo-labeled images from [37]. A flowchart explaining the pseudo-labelling technique [42] can be found in figure B.2.

In [10], a performance evaluation was carried out for each stage of the framework - localization and classification. For both, the corresponding dataset was split into 80% training data and 20% validation/testing data. To ensure a similar class distribution between training and test data, this split was done by preserving the percentage of samples for each class. Three different neural networks were analyzed for the localization task, namely EfficientDet [4], DETR [43] and Mask R-CNN [20]. The first two were trained to detect objects based on bounding-boxes, while the last relied on segmentation masks. One of the trained models was an EfficientDet-D2 with an EfficientNet-B2 backbone. Two different versions of DETR were trained, one with a ResNet-50 backbone and another with a ResNet-101 backbone. Lastly, the Mask R-CNN model used a ResNet-50 backbone. These four different models were trained and tested on the detect-waste dataset, considering only one class. As presented in table 3.2, the average precision of waste detection ranged from 28.0% for Mask-RCNN to 65.5% for EfficientDet-D2. Due to the significant gap between EfficientDets and the other architectures that were tested, the authors decided to limit the remaining experiments to this network family. In addition, after testing an EfficientDet-D3 the results were similar, so they decided to proceed with the previous network since it is smaller and requires less computational power.

Table 3.2: Accuracy results of selected architectures on detect-waste with one class - litter [10]

Model	Backbone	AP@0.50
DETR	ResNet-50	50.7
DETR	ResNet-101	51.6
Mask R-CNN	ResNet-50	28.0
EfficientDet-D2	EfficientNet-B2	65.5

Another experiment was made using EfficientDet-D2 to further understand its performance in different contexts - indoor, underwater and natural/urban. The datasets with indoor backgrounds, which include Drink-waste and MJU-Waste, presented the highest mAP@0.50 values with 99.4% and 97.9%, respectively. They were followed by TrashCan, which reached 91.3% with eight underwater waste classes. However, the overall worst score (7.3%) came from the other dataset with the same background type - Trash-ICRA. On the other hand, the datasets containing waste found "in the wild" - Extended TACO, UAVVaste and Wade-AI - achieved an accuracy of 56.8%, 71.5% and 74.1%, respectively. The results are presented in table 3.3.

Table 3.3: Results on different datasets using EfficientDet-D2 [10]

Background	Dataset	N. of Classes	mAP@0.50
Natural/Urban	Extended TACO	1	56.8
	Wade-AI	1	71.5
	UAVVaste	1	74.1
Underwater	Trash-ICRA	7	7.3
	TrashCan	8	91.3
Indoor	MJU-Waste	1	97.9
	Drink-waste	4	99.4

During the last experiment mentioned in [10], the previous model was trained and tested in four different configurations of datasets - Extended TACO (with one and seven classes), detect-waste and detect-waste+ (both with one class). In this case, a few evaluation metrics were added to have a more detailed understanding of the model's performance, such as mAP@0.50:0.95, mAP@0.75, AP_S, AP_M and AP_L. As presented in table 3.4, the highest accuracy for all ranges of IoU is achieved with detect-waste+, which also had the highest score for medium (AP_M) and large (AP_L) object instances. The Extended TACO with seven classes had the poorest performance, only achieving 16.2% of mAP@0.50. It is also mentioned that with this specific dataset a mAP@0.50 of 43.3% was achieved for the "metals and plastic" class, while for the remaining six classes it varied between 0.1% for "bio" and 8.9% for "unknown litter". The highest AP_S (19.8%) was achieved using the same dataset, but only with one class. Furthermore, there was a significant improvement of 3.4 percentage points in AP_S between detect-waste and detect-waste+, due to the additional data from [35].

Table 3.4: EfficientDet-D2 final evaluation results [10]

Dataset	Classes	mAP@0.50	mAP@0.75	mAP@0.50:0.95	AP _S	AP _M	AP _L
Ext. TACO	7	16.2	13.0	11.9	6.4	9.4	15.0
Ext. TACO	1	56.8	43.5	40.4	19.8	37.2	51.6
detect-waste	1	65.5	50.2	45.8	5.9	49.1	59.7
detect-waste+	1	66.4	51.3	46.9	9.3	51.3	59.8

In the classification stage, two different architectures were chosen - ResNet-50 and EfficientNet-B2. Both networks output the seven classes of waste previously mentioned; or eight classes, which include an additional "background" class to reduce the amount of false-positives. To tackle the uneven class distribution, two different types of samplers were used - random and weighted - where the weighted sampler provides more balanced data for each class. As mentioned above, the classify-waste+ dataset contains several pseudo-labeled images from [37]. Therefore, the authors of [10] also used a "pseudo-labelling type" hyperparameter to evaluate the model's performance, which shows if the pseudo-labels were updated every batch, every epoch or not at all. During training, several data augmentation approaches were used, such as applying rotation, gaussian blur and random brightness. Both training and testing datasets were also resized to the standard input size of EfficientNet-B2, 224x224. However, the authors hypothesize that augmentation did not significantly improve the results because of the natural diversity of the datasets used. The results in table 3.5 show a significantly better classification performance from EfficientNet in comparison to ResNet. The highest score was 73.0% for EfficientNet-B2, while the lowest was 60.6% for ResNet-50. They also concluded that updating pseudo-labels in every batch might slightly increase the accuracy.

Table 3.5: Accuracy results of selected classifiers (Adapted from [10])

Model	N. of Classes	mAP	Sampler	Pseudo-labeling
EfficientNet-B2	8	73.0	Weighted	per batch
EfficientNet-B2	8	74.6	Random	per epoch
EfficientNet-B2	8	72.8	Weighted	per epoch
EfficientNet-B2	7	72.7	Random	per epoch
EfficientNet-B2	7	68.3	Weighted	per epoch
EfficientNet-B2	7	74.4	Random	None
ResNet-50	8	60.6	Weighted	None

Considering the model with the best score with a weighted sampler (73.0%) in the previous experiment, a more detailed view of its performance for each waste class is presented in table 3.6. The "metals and plastic" class was predicted with the highest precision of 87%, due to its large representativity inside the dataset. On the other hand, a low precision of 52% is achieved for the detection of "unknown" and "non-recyclable" classes. The high F1-score value for "glass" shows the balance between false positives and false negatives. In addition, the background was successfully distinguished from the objects with a precision and recall of 97%.

Table 3.6: Summary of the precision, recall and F1-score for each waste class [10]

Class	Precision	Recall	F1-score
Background	0.97	0.97	0.97
Bio	0.62	0.51	0.56
Glass	0.83	0.82	0.83
Metals and Plastic	0.87	0.70	0.78
Non-recyclable	0.52	0.65	0.58
Other	0.71	0.74	0.72
Paper	0.62	0.76	0.68
Unknown Litter	0.52	0.65	0.58

3.4 Mask R-CNN with TACO

The authors of TACO [11] also did some experiments using a Mask R-CNN model trained on their dataset. This model used a default ResNet-50 in a Feature Pyramid Network as the backbone, considering an input layer size of 1024x1024. To assess its performance in waste detection and segmentation, they relied on two different dataset configurations: TACO-1, where there is only one class - litter; and TACO-10, which is divided into 10 super-categories, as presented in figure B.7. The weights that were initially used to train Mask R-CNN, were pre-trained on the COCO dataset following the steps in [44]. Due to the small amount of data in TACO, the results were obtained using a 4-fold cross validation, where for each fold the dataset is split randomly into 80% training, 10% validation and 10% testing data. They relied on AP over different IoU thresholds as the evaluation metric. As demonstrated in [45], "the mask quality, quantified as the IoU between the instance mask and its ground truth, is usually not well correlated with classification score", so it was proposed a network block that takes the instance feature along with the predicted mask to obtain a more accurate mask IoU. Following this method, three different scores were tested in [11]. Assuming the class probabilities $P = p_1, p_2, \dots, p_{N+1}$, such that N is the number of classes and p_{N+1} is the probability of being background, the following scores were used:

$$Score = \begin{cases} \max_i p_i, & class_score \\ 1 - p_{N+1}, & litter_score \\ \frac{\max_i p_i}{p_{N+1} + \epsilon}, & ratio_score \end{cases} \quad (3.1)$$

The standard mAP metric corresponds to *class_score*, while *litter_score* expresses the probability of not being part of the background and *ratio_score* defines both the confidence in a class and in being litter (the ϵ is just a small scalar to avoid NaN values). As expected, looking at the results in table 3.7, the typically used *class_score* is not as suitable for TACO-1 as the *litter_score*, which achieved 26.2%. On the other hand, the *ratio_score* was the highest for TACO-10, with 19.4%. In addition, the confusion matrices in figure B.8 show a better detection performance for the classes "bottles" and "cans", while there is some confusion between "plastic bag" and "other".

Apart from the "other" class, the "cigarette" class was the one with the highest amount of false negatives.

Table 3.7: AP for 4-fold cross validation using different scores (%) to rank predictions [11]

Dataset	class_score	litter_score	ratio_score
TACO-1	15.9 ± 1.0	26.2 ± 1.0	26.1 ± 1.0
TACO-10	17.6 ± 1.6	18.4 ± 1.5	19.4 ± 1.5

3.5 Faster R-CNN

The work in [12] presents a different approach to waste classification using two different configurations of a Faster R-CNN model - one with a ResNet-101 and another with an Inception-V2 [46] backbone. The Inception network family was developed soon after ResNet and both share a similar scaling principle, in the sense that Inception scales by width instead of depth. In this case, they are both used for feature extraction.

The dataset used is a combination of Google search using queries such as "roadside waste", specifying the required type of litter; a few pictures collected by volunteers using smartphone cameras; and even some images from the TrashNet dataset. This compilation resulted in a dataset of 1013 images with ten different classes of litter and more than 3000 instances in total (see table 3.8). Some data augmentation techniques were also applied, such as image resizing to 800x600, image rotation and flip.

Table 3.8: Number of images and instances per class [12]

Class	N. of instances	N. of images
bottle	235	187
cigarette	245	79
can	239	137
cardboard	416	176
cup	138	59
gloves	139	58
leaf	707	150
mask	108	72
plastic	729	127
vegetation	74	68

For the training process, the dataset was split between 75% training data, 10% validation data and 15% testing data. Both architectures were pre-trained using COCO data. The best performance is achieved with Inception-V2 - a mAP of 92% (see table 3.9). On the other hand, the ResNet-101 had an accuracy of 83%. The results of the average accuracy for each class are presented in table 3.10, where a more even score distribution is also achieved with Inception rather than with ResNet.

Table 3.9: Classification performance for Faster R-CNN models (Adapted from [12])

Model	mAP
Faster R-CNN (Inception V-2)	0.92
Faster R-CNN (ResNet-101)	0.83

Note that in the following table an 11th class is presented - "paper" - which is never mentioned in [12]. This additional class is assumed to be a mistake since it is never mentioned in [12]; and since the mAP calculated in table 3.9 did not take it into consideration.

Table 3.10: Accuracy results for each class using Faster R-CNN models (Adapted from [12])

Class	Faster R-CNN (Inception V-2)	Faster R-CNN (ResNet-101)
bottle	0.99	0.98
cigarette	0.87	0.80
can	0.99	0.93
cardboard	0.90	0.75
cup	0.92	0.50
gloves	0.95	0.98
leaf	0.86	0.67
mask	0.93	1.00
paper	0.78	0.63
plastic	0.88	0.72
vegetation	0.91	1.00

Chapter 4

State-of-the-art Analysis

In this chapter, the results reported in chapter 3 will be further analyzed and compared, considering this project's goal - detecting objects in an urban context. The advantages and limitations of each solution will be discussed, mainly focusing on the dataset's reliability, the model's accuracy, and the inference speed. Some properties of each model, reported in chapter 2, will also be taken into consideration.

4.1 Datasets

The dataset used in [6] includes some very relevant classes that might fit the goals of this project since its content may not be very different than the same classes of waste found in Portugal, such as "waste bag" and "cardboard". Nevertheless, this dataset is not publicly available, and 1400 labeled images are a very limited amount of data to train a model. Furthermore, the testing set used is only composed of 112 pictures, in which the amount of object instances corresponds to less than 7% of the ones found in the training dataset, which inevitably does not provide an accurate representation of each class.

In [10], the collected datasets - **detect-waste+** and **classify-waste+** - were the most diverse and complete for waste detection found during the literature review. Both provide a large amount of data, particularly for natural and urban contexts (a total of around 7000 images - see figure B.3), which aligns with the scope of this work. Another major advantage is the fact that all these datasets are publicly available. Composing detect-waste+, there is a particular dataset worth highlighting - **TACO** - whose main advantage is its diversity in waste categories and backgrounds. However, considering its crowd-sourcing nature, annotations may contain some user-induced errors and biases, and the small amount of annotated data (about 1500 images) mentioned in [11] also represents a performance limitation. Moreover, these datasets have very few annotations of the "waste bags" and "cardboard" classes (particularly with a street or sidewalk background), if not none.

On the other hand, the dataset used in the last application presented in chapter 3 is very small (just over 1000 images) and random since it relies on several different sources, including Google

searches. Despite the very promising results achieved with Faster R-CNN, they might include some overfitting because of this limitation.

This project requires a large number of object instances in as many urban backgrounds as possible, always ensuring balance across categories. Even though the dataset used in ODK has the most similarities with this project's goals, it is very limited and publicly unavailable. On the other hand, datasets such as TACO (or Extended TACO), Wade-AI, and UAVVaste are open-source and contain a wide variety of objects and backgrounds in similar contexts. Nonetheless, none of them includes a significant amount of annotations of "waste bags" and "cardboard" that are similar to the ones expected to find on the streets of Madeira.

4.2 Accuracy

Regarding the accuracy of each approach, the mAP@0.5 of 61% obtained with ODK is very similar to the value reported in [23] for a YOLOv3 model trained and tested on the COCO dataset - 57.9%. Furthermore, by looking at the mAP values for each class in table 3.1, it is clear that the objects with more training data and/or homogeneous appearance have the highest scores, such as the "small container", "pole" and "garbage bag" classes. Additionally, the "graffiti" class achieved the best precision (0.47%), which may be related to the fact that this type of data is very easily distinguished from the environment. Without any further details, overfitting may also be one of the reasons. On the other hand, less homogeneous classes - "cardboard" - and with very little data - "mattress", had the worst performances, as expected. Thus, it might be relevant to further train this model with a larger dataset to correctly assess its performance. It is also worth mentioning that this dataset is not publicly available, so a similar and larger dataset must be selected for this task.

As mentioned in the previous chapter, the "Detect Waste in Pomerania" project is divided into two different tasks - localization and classification. For the localization stage, the Extended TACO, Wade-AI, and UAVVaste datasets were the ones with the worst mAP@0.50, as seen in table 3.3. However, the average precision for Extended TACO (56.8%) is the only one that doesn't surpass the precision of an EfficientDet-B2 trained and tested on COCO (62.7%), reported in [4]. One of the reasons for this to happen might be the lack of balance between classes in this dataset (see figure B.4). Moreover, the fact that 45% of its instances are very small (less than 32x32 pixels) might also be affecting its performance, as mentioned in [10]. Considering the datasets with natural/urban background - Extended TACO, Wade-AI, and UAVVaste - the average precision results (56.8%, 71.5%, and 74.1%, respectively) are well above the 61% reported with ODK, also with the exception of the Extended TACO. On the other hand, the datasets with the plainest and most homogeneous backgrounds - Drink-waste and MJU-waste - achieved the best scores, confirming the impact of context in object detection data.

The experiments carried out in this work show a better performance for object localization using EfficientDet, when compared to other architectures based on ResNets. Similarly, ResNet was again defeated in the classification stage - this time by an EfficientNet, which is used in the

backbone of EfficientDets. This leads to the conclusion that EfficientNet-based models might be a better approach to object detection tasks in comparison to ResNet-based ones. However, even though ResNet's reported accuracy appears to be much lower than the worst score with EfficientNet (see table 3.5), it should be noted that ResNet-50 was only tested once and without any pseudo-labeling update, while EfficientNet-B2 was tested with six different configurations. Therefore, it can be relevant to perform more tests using the ResNet to come to a more solid conclusion. One could also test a more complex version of EfficientDet to assess if its accuracy increases, such as EfficientDet-D4. Nevertheless, the authors of [10] also performed some classification tests on an EfficientNet-B4 that, in fact, achieved worse accuracy than EfficientNet-B2. Since this network is used as the backbone of EfficientDet-D4, this might mean that there will be no improvements with this model upgrade.

Regarding the EfficientNet results in table 3.6, the high precision of the class "metals and plastic" is due to its large representativity (see figure B.5), as previously mentioned. On the other hand, the "non-recyclable" and "unknown litter" classes are the ones with the lowest score, which might be connected to the shape and size diversity of its data. In addition, the confusion matrix in figure B.6 shows that these two classes are mostly confused with the "metals and plastic" class, which is expected since its training data far exceeds the rest.

The approach described in [11] using Mask R-CNN resulted in a very low accuracy, as reported in table 3.7. This is mainly related to the fact that training a model with segmentation masks is a lot more challenging than with bounding boxes since the annotation task is much more time-consuming and, consequently, there is not as much annotated data available. In addition, the data required for object detection "in the wild" often contains small object instances, for which Mask R-CNN does not show a good performance. In fact, there is a poor "cigarette" detection observed in the confusion matrices B.8, where this class is mostly confused with the background. This might happen due to the small size of the data, shown in figure B.7, which had to be resized to less than one-third (1024x1024 px) to match the model's requirements. A slight confusion between the classes "plastic bag" and "other" is also noticeable, which is expected since the "other" class data includes objects with similar materials but presented in different shapes and sizes. To improve performance, the authors of [11] also suggest a "transplantation" approach to augment the dataset - this method takes advantage of very precise segmentation masks to place specific objects on under-represented backgrounds artificially. However, there is always the risk of losing some important information, such as the object's lighting conditions. Future work with this dataset should rely on models that fully exploit the high resolution of its images, and it also should be more focused on training with bounding boxes instead of masks in order to achieve better results.

Lastly, in [12], the standard backbone of the Faster R-CNN architecture - VGG16 - is replaced with more accurate networks - ResNet-101 and Inception-V2. Even though these achieve higher accuracy, their greater complexity inevitably leads to slower object detection. In fact, as mentioned in chapter 2, a Faster R-CNN model with a ResNet-101 achieves a very similar accuracy (83.8%) as the one reported in this paper (83%). However, despite reporting better results than the previous approaches, there is a very high probability of overfitting due to the very limited dataset it was

trained and tested on.

As in any machine learning application, if the training is done with a large amount of balanced data, the higher the model's accuracy, the better the solution. Thus, considering the previous discussion, it appears that the YOLO-based model is the least accurate among the proposed models. On the other hand, the Faster R-CNN approach has the best results. However, the limited training and testing process doesn't assure the same results on a larger scale. In between those two remains the EfficientDet/EfficientNet model with relatively high accuracy, considering a much larger dataset than the others.

4.3 Real-time Detection

Regarding detection speed, the YOLOv3 model can achieve its best accuracy value at around 20 FPS [23]. For ODK, this speed is enough to perform detection in real-time because the data is only collected to the server at a rate of 3 frames per second [6]. Considering the YOLOv5's superior performance mentioned in [28], better results should be achieved by using this model instead of YOLOv3.

In [10], the model's speed is never mentioned, but according to [4], the EfficientDet-D2 can perform at around 50 FPS (on a V100 GPU), which is faster than the most accurate version of YOLOv3 and similar to the results reported for YOLOv5. The authors of TACO also don't mention the inference speed of the model they used. However, according to [20], a Mask R-CNN with a ResNet-101-FPN can run at 195 ms per image (approximately 5 FPS) on a Nvidia TeslaM40 GPU. The model mentioned in [11] uses a ResNet-50-FPN backbone, which is less complex than the previous one, so its inference speed should be as fast or even faster. Regarding the last approach, no other Inception-V2 model applied to waste detection was found, so the reference test speed was estimated based on the results achieved in [47, 48], which mention values between 8 and 21 FPS, depending on the image input size. On the other hand, [12] reports that despite the model being 10.6% more accurate with a ResNet-101 than with a VGG16 (73.2%), the test speed is also over 20 times lower - approximately 0.4 FPS. This leads to the conclusion that in order to achieve faster performance, a less complex architecture should be used, such as ResNet-50 or EfficientNet-B2.

Chapter 5

Object Detection in the Public Landscape

This chapter is divided into three main sections - "System Architecture", "Detection of People and License Plates", and "Detection of Waste, Graffiti, Traffic Signs, and Billboards". In "System Architecture" the project's layout is displayed, as well as how every part of it operates and connects with each other. The last two sections represent the two main tasks this project aimed to achieve - training a model to detect people and license plates and another one to detect waste, graffiti, traffic signs, and billboards in a public landscape. These are divided into four different subsections - "Datasets and Preprocessing", "Models", "Training Results", and "Summary". In "Datasets and Preprocessing", every dataset used to train the models and any data cleaning or modification that was needed will be mentioned in this subsection; in "Models" it will be discussed which models (and which versions of those models) were used for each task; in "Training Results" every training that was done during this project and its corresponding results will be thoroughly described in chronological order; lastly, in "Summary" the most relevant information from the previous three subsections of that task will be summarized.

5.1 System Architecture

Similar to the "Object Detection Kit" mentioned in chapter 3, the architecture of this project consists of a mobile device that will be placed in logistics company's vehicles to record the surroundings while it drives around the city, using a mobile application developed for this purpose; a server where the frames of the previous recordings will be processed and annotated; another server where the annotated frames will be transferred to and stored; and a web interface where the responsible staff will be able to check the location and object class of every detected object in a city map (see figure 5.1). This dissertation will focus on the training of the computer vision models.

After the literature review from chapters 2 and 3, the work plan for the models' training process was divided into two main tasks. For the first one, a model was trained to run on the mobile device and detect people and license plates in real-time, to later blur them from every frame before they

were transferred to the server. The blurring is required to ensure there is a privacy filter since the pictures are taken without any consent from the citizens. This way, the images are already sent to the server censored and ready to be saved. Afterwards, another model was trained to run on the server and detect different objects in public spaces, such as waste containers, waste bags, cardboard boxes, traffic signs, graffiti, and billboards. Every training session that will be discussed in this chapter was done on a Nvidia GeForce RTX 3090 GPU, in a computer also with an 11th Gen Intel(R) Core(TM) i9-1199F @ 2.5GHz (8 cores) CPU.

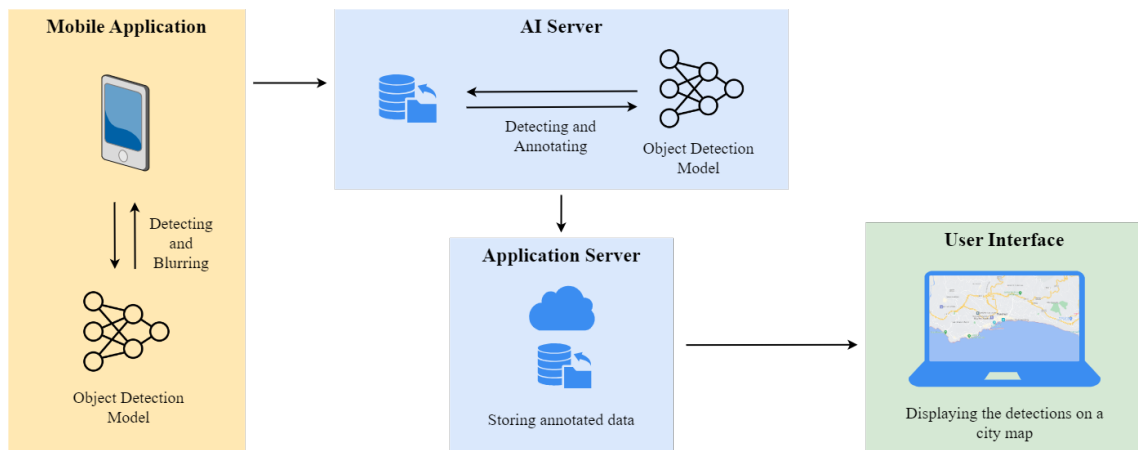


Figure 5.1: System Architecture

5.2 Detection of People and License Plates

The first approach to completing this task was to use an open-source project that already had trained a model to detect these two classes. This approach should avoid needing to train a model from scratch. However, after searching for such a project on platforms like GitHub, there were not a lot of viable options. Most of them were either outdated, didn't detect both classes simultaneously, or didn't use a model as simple and easy to use as YOLO. Ultimately, it was decided that a YOLO model - YOLOv7 [24], in particular - would be trained to fulfill our specific needs. A YOLO model was chosen because of its lower computational requirements, which aligns with the ultimate goal of this task - to run the model on a mobile device. On the other hand, YOLOv7 was chosen instead of the most recent version - YOLOv8 - since this last one was only released after this project started being developed.

5.2.1 Datasets and Preprocessing

None of the datasets mentioned in chapter 4 included the required classes of objects for this task - people and license plates. As a solution, the staff of a logistics company from Funchal, Madeira, took several pictures of their work travels throughout the city from the passenger's point of view.

At a rate of 4 FPS, they collected more than 16000 images that were stored in a Label Studio [49] database, where object detection models could also be tested. However, after analyzing them, it was concluded that manually annotating each instance of people and license plates in this dataset would be very time-consuming. While searching for new datasets online that could solve this problem, the "Open Images" [50] dataset was found. This dataset was developed by Google and has around 1.9 million images with 16 million bounding boxes for 600 different classes, including "Person" and "Vehicle registration plate". Even though this dataset also included a "Human face" class, the "Person" class with full-body annotations was chosen to provide a more robust privacy filter. In addition to the "Open Images" dataset, some images from the MS COCO "Person" class were also collected. Since "Open Images" is a very large dataset with many different classes, a tool called "FiftyOne" [51, 52] (developed by the Voxel51 company) was used to download a specific amount of images, while only selecting the desired classes. Roboflow [53] was also used to edit, add and remove annotations from the datasets. The next sections explain the use of these datasets in detail.



Figure 5.2: Example images from the datasets used for this task

5.2.2 Models

As previously mentioned, the model chosen to train for this task was the YOLOv7 from the GitHub repository [54], which has the official implementation of the corresponding paper [24]. This repository includes several different weights for different architectures of this model that were trained on the MS COCO dataset. By ascending order of complexity, the most relevant ones are the YOLOv7-tiny, YOLOv7, YOLOv7-x, YOLOv7-w6, YOLOv7-e6, YOLOv7-d6, and YOLOv7-e6e. Considering the goal of implementing this model using edge computing, the YOLOv7-tiny was the first one to be trained because of its small size in comparison to the others. According to [24], the mAP@0.5 for this particular model is around 47.3% (for a 320x320 input size) and 52.8% (for a 416x416 input size) when trained on MS COCO dataset. After achieving poor results with it, training sessions with the YOLOv7-e6 and YOLOv7-e6e followed, which will be displayed in the next section 5.2.3. For the YOLOv7-e6, an expected mAP@0.5 would be around 73.5% for an input size of 1280x1280, and for the YOLOv7-e6e with the same input size, a mAP@0.5 of 74.4% would be expected (as described in table 5.1).

Table 5.1: State-of-the-art accuracy results using YOLOv7 models, by ascending order of complexity

Model	Input Size	mAP@0.50
YOLOv7-tiny	320x320	0.473
YOLOv7-tiny	416x416	0.528
YOLOv7-e6	1280x1280	0.735
YOLOv7-e6e	1280x1280	0.744

5.2.3 Training Results

As mentioned, all the following training results were obtained with a RTX 3090 GPU using the pre-trained weights from the repository [54]. The input size chosen for each training depended on the model, and it was the same as the input size mentioned in Tables 1 and 2 of the YOLOv7 paper [24] (except for the YOLOv7-tiny, which was trained using an input size of 640x640 to reduce the loss of image quality). Moreover, all the training sessions mentioned below, with the exception of training 12 and 12.1, used the hold-out cross-validation technique, mainly with around 70% for the training set and 30% for the validation set. The last two used k-folds cross-validation (with five folds). Finally, some of the images recorded in the streets of Funchal were used as the testing set to assess each model's performance. These images have the corresponding people and license plates manually censored to ensure privacy.

The first training session was done with YOLOv7-tiny on the "Open Images" dataset. Exactly 3000 images, including the classes "Person" and "Vehicle registration plate" were downloaded from this dataset. These images were split into 2400 for the training and 600 for the validation sets. Using an input size of 640x640 and training for 100 epochs, the mAP@0.5 was 0.431 (0.198 for people and 0.665 for license plates), and the mAP@0.5:0.95 was 0.272. Since the dataset's size

could be a reason for the lack of performance, another training was done using around four times the number of images (10000 for training and 2000 for validation). These images also included other "Open Images" classes that were added to the "Person" class to improve performance - "Man" and "Woman". Nevertheless, the results remained very poor after 50 epochs (with a mAP of 0.519 - 0.430 for people and 0.609 for license plates). So a different approach was taken - training only to detect people and then retraining the same model to detect license plates. For the next training session, the dataset consisted of around 4500 images from the "Person" class of the MS COCO dataset, which is a well-known benchmark dataset. After 50 epochs, the mAP@0.5 was 0.654, and the mAP@0.5:0.95 was 0.420. Finally, this model was retrained using the best weights from the previous training. Also, around 4500 images with license plates from the "Open Images" were added to the previous dataset, which led to a mAP@0.5 of 0.693 (0.607 for people and 0.780 for license plates). The overall results are displayed in table 5.2.

Table 5.2: Accuracy results using the YOLOv7-tiny model

Training	Dataset ¹	Split (Train/Val)	mAP@0.50	mAP@0.50:0.95
1	Open Images (P;L)	2400/600	0.431	0.272
2	Open Images (P;L)	10000/2000	0.519	0.35
3	COCO (P)	3488/987	0.654	0.420
4	COCO (P) + Open Images (L)	6946/2036	0.693	0.401

Considering the YOLOv7-tiny training with the best results - training 4 in table 5.2 - the validation set consisted of 2036 images with 4296 annotations of people and 1512 annotations of license plates. Running on these validation images, the inference time was 1.8 ms with GPU and 67 ms with CPU - using a batch size of 32.

Even though confusion matrices are not very reliable for object detection because of their dependency on the confidence threshold selected (as mentioned in chapter 2), these can visually help understand a model's performance in more detail. In this case, figure 5.3 shows the person class was correctly predicted 70% of the times, while it was confused with the background in the remaining 30%. On the other hand, 87% of the annotations with license plates were correctly predicted, while the remaining 13% were confused with the background (considering a confidence threshold of 0.25).

A few important details should also be addressed about this confusion matrix and the ones that will follow in this chapter. On one hand, these matrices are automatically generated after running the "train.py" script in [54], and "background FN" and "background FP" are added even though there was no "background" class in the validation set. The row named "background FN (False Negatives)" might be a little confusing to understand since it refers to the background's False Positives (FP) and not the False Negatives, as the row's name suggests. In fact, the values on this row represent the False Negatives from the person and license plate classes. On the other hand, the

¹P: represents the "Person" class; L: represents the "License plate" class

values on these confusion matrices are normalized, which means they don't show the actual number of correct and incorrect detections for each class, only the percentage of all detections for that class. Hence, even though the last column ("background FP") shows that 83% of "background" detections were confused with the person class, it doesn't affect the overall performance of the model since it is a large percentage of a very small amount of detections (which would be even smaller if a greater confidence threshold was chosen). In addition, some images from Madeira were used as the testing set of this model, and a few results can be seen in figure 5.4.

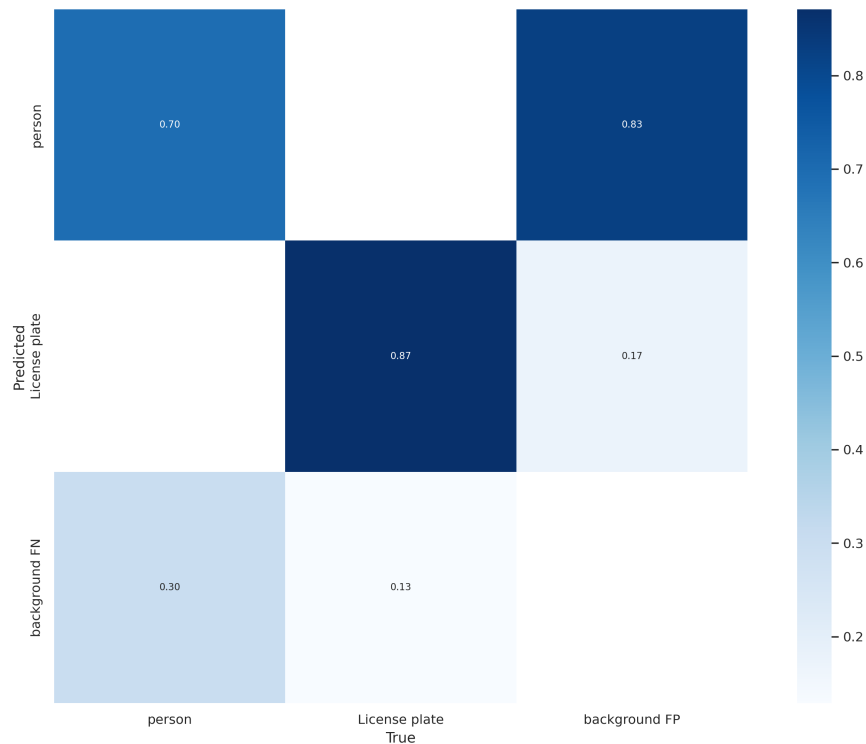


Figure 5.3: Confusion matrix of training 4 in table 5.2

The next training sessions were made using the YOLOv7-e6 model, always with an input size of 1280x1280. The first one relied on the same dataset as the last training with YOLOv7-tiny, which included images with people from MS COCO and with license plates from "Open Images". Yet, since this model requires more computational power, the dataset was reduced to 2000 images for each class instead of 4500. For the same reason, the batch size also had to be reduced from 32 to 8. After 50 training epochs, the mAP@0.5 was 0.721 (0.624 for people and 0.818 for license plates) and the mAP@0.5:0.95 was 0.465. This model was then retrained for 50 epochs, achieving a mAP@0.5 of 0.726 (0.638 for people and 0.818 for license plates) and a mAP@0.5:0.95 of 0.464. To confirm if a bigger dataset could improve its performance for people detection, another 3000 images with people were added to the dataset, combining a total of 7000 images. The following training appeared to show some accuracy improvements, with a mAP@0.5 of 0.757 (0.689 for people and 0.826 for license plates) and a mAP@0.5:0.95 of 0.495. However,



Figure 5.4: Testing images annotated using training 4 in table 5.2 (yellow bounding boxes for "license plate" and pink for "person")

when tested on the testing set, the people detection remained very poor.

After analyzing the results and the datasets, it was concluded that the images with license plates also included people that were not annotated, which was the main reason for the model's lack of performance. Hence, the first step was training a model to detect people and then using it to annotate the images in question. The model trained with 5000 people images from COCO had a $mAP@0.5$ of 0.730 and a $mAP@0.5:0.95$ of 0.539. Around 2000 images from "Open Images" were annotated with the previous model, and a new dataset was created, combining these images with 3000 images from COCO. This time, the $mAP@0.5$ and $mAP@0.5:0.95$ were 0.788 (0.752 for people and 0.825 for license plates) and 0.515, respectively. Finally, considering the accuracy improvements that resulted from this correction, the last model was then retrained for another 50 epochs, achieving a $mAP@0.5$ and a $mAP@0.5:0.95$ of 0.797 (0.761 for people and 0.833 for license plates) and 0.515, respectively.

Table 5.3: Accuracy results using the YOLOv7-e6 model

Training	Dataset ¹	Split (Train/Val)	$mAP@0.50$	$mAP@0.50:0.95$
5	COCO (P) + Open Images (L)	2788/1212	0.721	0.465
5.1	COCO (P) + Open Images (L)	2788/1212	0.726	0.464
6	COCO (P) + Open Images (L)	4849/2151	0.757	0.495
7	COCO (P)	3450/1550	0.730	0.539
8	COCO (P) + Open Images (L)	3450/1550	0.788	0.515
8.1	COCO (P) + Open Images (L)	3450/1550	0.797	0.515

Considering the YOLOv7-e6 training with the best results - training 5.1 in table 5.3 - the validation set consisted of 1550 images with 4892 annotations of people and 871 annotations of license plates. Running on these validation images, the inference time was 17.8 ms with GPU and 1659 ms with CPU - using a batch size of 4. In this case, figure 5.5 shows the person class was

¹P: represents the "Person" class; L: represents the "License plate" class

correctly predicted 81% of the times, while in the remaining 19% it was confused with the background. On the other hand, 90% of the annotations with license plates were correctly predicted, and only 10% were confused with the background (considering a confidence threshold of 0.25). In addition, some images from Madeira were used as the testing set of this model, and a few results can be seen in figure 5.6.

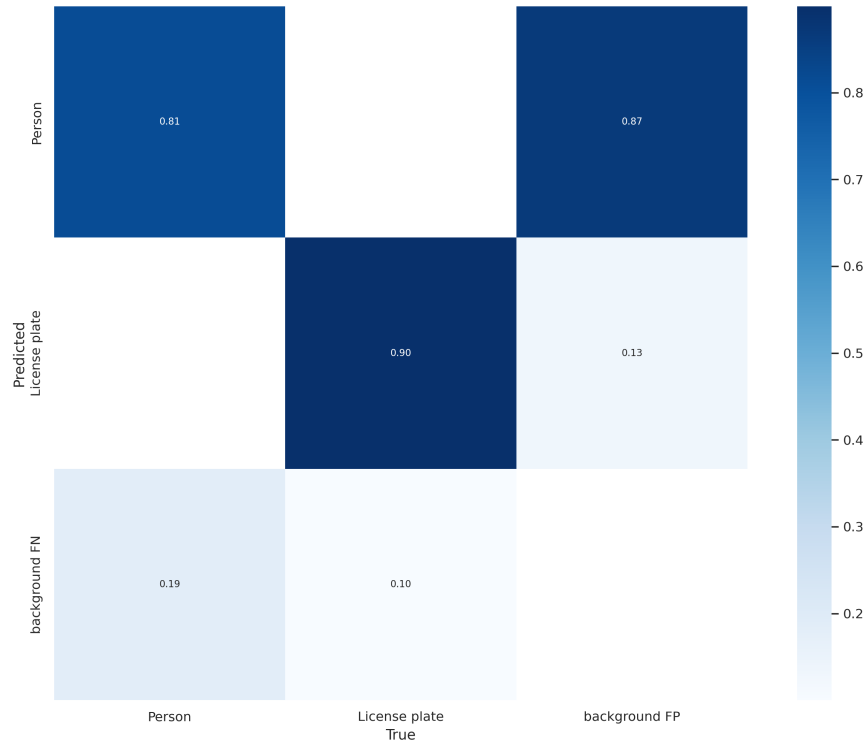


Figure 5.5: Confusion matrix of training 5.1 in table 5.3



Figure 5.6: Testing images annotated using training 5.1 in table 5.3 (green bounding boxes for "license plate" and yellow for "person")

Following the accuracy improvements of training 5 and 5.1 in table 5.3, the same dataset of 5000 images was then used to train a YOLOv7-e6e model. Once again, because of the increasing complexity between models, the training batch size was reduced to 4. The first training resulted in a mAP@0.5 of 0.793 (0.751 for people and 0.835 for license plates) and a mAP@0.5:0.95 of 0.518. However, training the same model for another 50 epochs didn't improve the performance, only achieving a mAP@0.5 of 0.786 (0.742 for people and 0.830 for license plates) and a mAP@0.5:0.95 of 0.508. The tests done on the testing set still showed some incorrect detections of people, such as detecting a group of people as one single detection, detecting specific body parts as a person, or detecting people that are very far away and can't be recognized. Since these particular detections are irrelevant to this application, all 3000 images from COCO were uploaded to Roboflow, and the annotations considered irrelevant were manually edited or removed. In the end, only 2794 images remained. The model was again retrained with the new dataset, which resulted in an even lower accuracy. The mAP@0.5 was 0.767 (0.733 for people and 0.801 for license plates) and a mAP@0.5:0.95 of 0.506. Besides not detecting the group of people mentioned above anymore, there were no more visible improvements.

Considering the accuracy values of the previous models, the next approach to improve performance even further was to compile a batch of 1000 images taken in Madeira, use the model from training 5.1 (in table 5.3) to annotate these images, upload them to Roboflow and manually edit the necessary annotations, and retrain the model using the corrected images as the dataset. After editing, only 736 images remained from that batch. For this training session, the mAP@0.5 and mAP@0.5:0.95 were 0.881 (0.781 for people and 0.980 for license plates) and 0.687, respectively. The abovementioned process was repeated twice with two more batches of 1000 images each. Batch 1 and 2 combined for 1201 images (the remaining ones didn't have any annotations) and achieved a mAP@0.5 of 0.967 (0.953 for people and 0.981 for license plates) and a mAP@0.5:0.95 of 0.755. Finally, combining batches 1, 2, and 3 (2201 images) resulted in a mAP@0.5 of 0.960 (0.929 for people and 0.991 for license plates) and a mAP@0.5:0.95 of 0.760.

Table 5.4: Accuracy results using the YOLOv7-e6e model

Training	Dataset ¹	Split (Train/Val)	mAP@0.50	mAP@0.50:0.95
9	COCO (P) + Open Images (L)	3450/1550	0.793	0.518
9.1	COCO (P) + Open Images (L)	3450/1550	0.786	0.508
9.2	COCO (P) + Open Images (L)	3335/1459	0.767	0.506
10	Batch 1	510/226	0.881	0.687
10.1	Batches 1 and 2	826/375	0.967	0.755
10.2	Batches 1, 2, and 3	1561/640	0.960	0.760

Considering the YOLOv7-e6e training with the best results - training 10.2 in table 5.4 - the validation set consisted of 640 images with 333 annotations of people and 559 annotations of license plates. Running on these validation images, the inference time was 20.1 ms with GPU and 2007 ms with CPU - using a batch size of 4. In this case, figure 5.7 shows the person class was

¹P: represents the "Person" class; L: represents the "License plate" class

correctly predicted 95% of the times, while in the remaining 5% it was confused with the background. On the other hand, 99% of the annotations with license plates were correctly predicted, and only 1% were confused with the background (considering a confidence threshold of 0.25). In addition, some images from Madeira were used as the testing set of this model and a few results can be seen in figure 5.8.

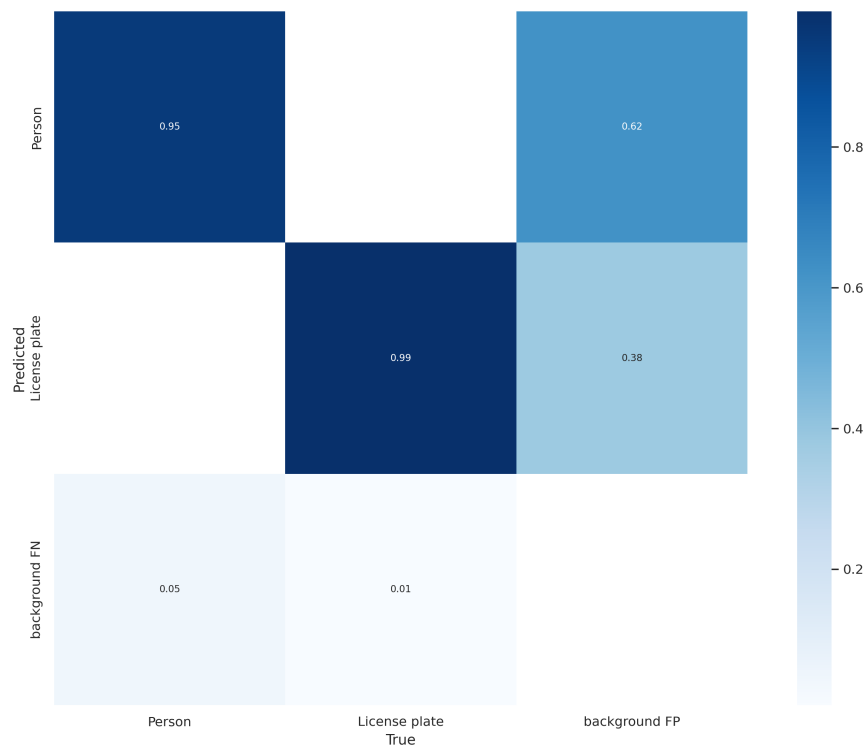


Figure 5.7: Confusion matrix of training 10.2 in table 5.4



Figure 5.8: Testing images annotated using training 10.2 in table 5.4 (pink bounding boxes for "license plate" and green for "person")

After improving the dataset along the training process and achieving the best accuracy results for this task with a training dataset only including annotated images from Madeira, one last experiment was done with YOLOv7-tiny to see which results it would get with this improved dataset. Relying on the weights from training 4 (in table 5.2), the overall mAP@0.5 and mAP@0.5:0.95 achieved with the YOLOv7-tiny was 0.788 (0.768 for people and 0.807 for license plates) and 0.530, respectively. Moreover, training the same model with another 50 epochs resulted in a mAP@0.5 of 0.813 (0.802 for people and 0.825 for license plates) and a mAP@0.5:0.95 of 0.551.

Considering the overall accuracy improvements achieved with the new dataset, the last training was repeated with k-folds cross-validation instead of hold-out, which was used for every training until this point. For this goal, a new Python script was developed to split the dataset into five folds and then train the model using each of these folds as the validation set while the remaining merged as the training set (see appendix C). To split the data, a function called "StratifiedKFolds" from sklearn library [55] was used. The reason for choosing this function was the fact that it maintains the overall ratio for each class in each fold when splitting the data, which should provide even more accurate results than the hold-out technique that splits the data randomly. After splitting, each fold had 441 images with 38.33% person and 61.67% license plate annotations. On average, the overall mAP@0.5 and mAP@0.5:0.95 for each fold after 50 epochs was 0.508 (0.416 for people and 0.666 for license plates) and 0.333, respectively. Finally, the previous model was retrained for 50 more epochs (each fold) to see if these values would improve. This time, the average mAP@0.5 and mAP@0.5:0.95 for each fold was 0.712 (0.640 for people and 0.784 for license plates) and 0.460, respectively.

Table 5.5: Accuracy results using the YOLOv7-tiny model with an updated dataset

Training	Dataset	Split (Train/Val)	mAP@0.50	mAP@0.50	Cross-validation
11	Batches 1, 2, and 3	1561/640	0.788	0.530	Hold-out (70/30)
11.1	Batches 1, 2, and 3	1561/640	0.813	0.551	Hold-out (70/30)
12	Batches 1, 2, and 3	1760/441	0.508	0.333	K-folds (k=5)
12.1	Batches 1, 2, and 3	1760/441	0.712	0.460	K-folds (k=5)

The validation set of the YOLOv7-tiny model with the best results in table 5.5 - training 11.1 - was exactly the same as in training 10.2. Running on these validation images, the inference time was 1.3 ms with GPU and 48.5 ms with CPU - using a batch size of 32. In this case, figure 5.9 shows the person class was correctly predicted 87% of the times, while in the remaining 13% it was confused with the background. On the other hand, 88% of the annotations with license plates were correctly predicted, and only 12% were confused with the background (considering a confidence threshold of 0.25). In addition, some images from Madeira were used as the testing set of this model, and a few results can be seen in figure 5.10.

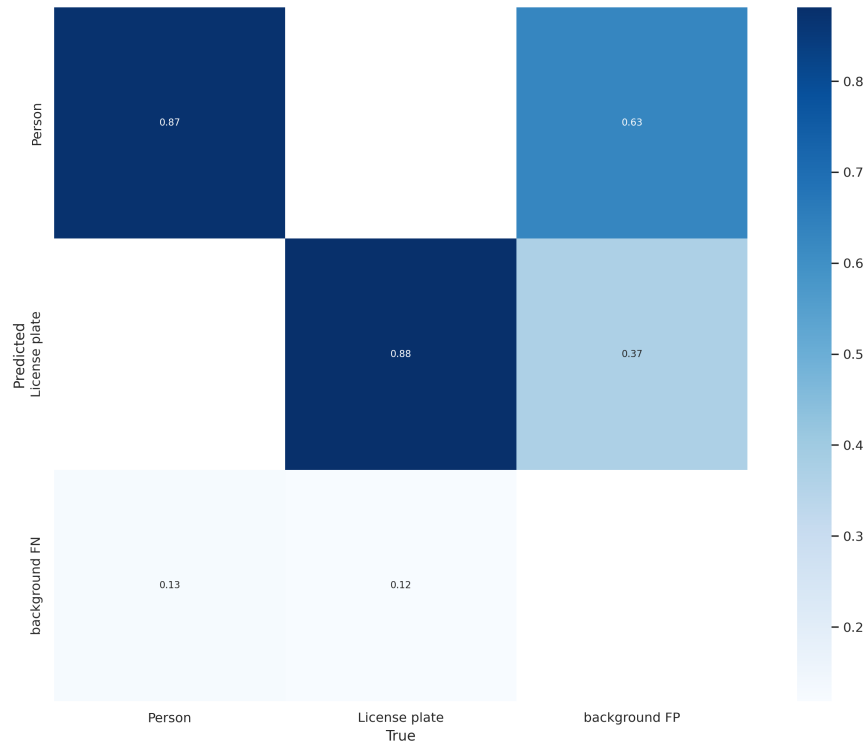


Figure 5.9: Confusion matrix of training 11.1 in table 5.5

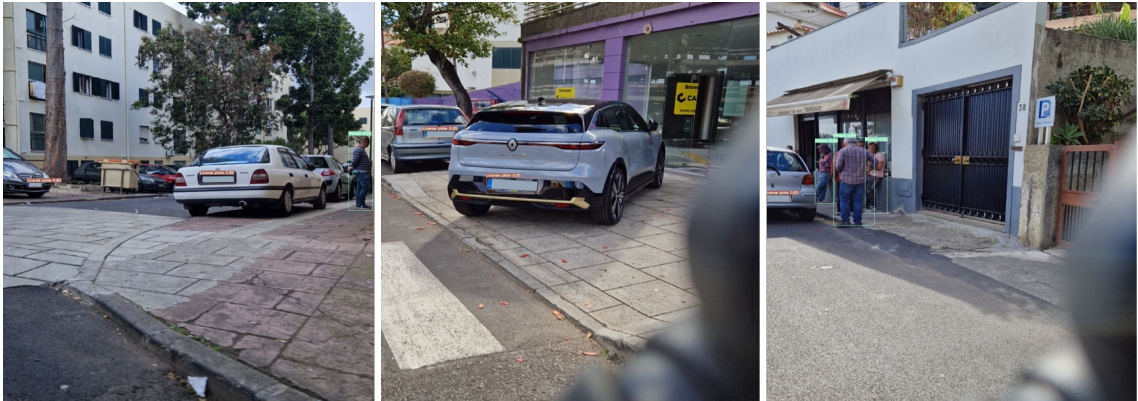


Figure 5.10: Testing images annotated using training 11.1 in table 5.5 (brown bounding boxes for "license plate" and green for "person")

5.2.4 Summary

A combination of three different datasets was used to train three different YOLOv7 models - YOLOv7-tiny, YOLOv7-e6, and YOLOv7-e6e - to detect two different classes, people, and license plates. After changing a few parameters, such as the ratio of training and validation images and the input size, editing some incorrect annotations, and using different cross-validation techniques, very different outcomes were achieved. The accuracy values increased along with the complexity

of each model, as expected. However, the best results were only achieved after using annotated images collected in Madeira as the training and validation dataset. The accuracy values per class for the best-performing models are displayed in table 5.6, and the inference time achieved with each YOLOv7 model version is summarized in table 5.7.

Table 5.6: Accuracy results for the best-performing models using annotated batches 1, 2, and 3 as training dataset - first task

Training	Model	Person		License Plate	
		mAP@0.50	mAP@0.5:0.95	mAP@0.50	mAP@0.5:0.95
12.1	YOLOv7-tiny ²	0.640	0.373	0.784	0.547
11.1	YOLOv7-tiny	0.802	0.497	0.825	0.605
9	YOLOv7-e6e ³	0.751	0.565	0.835	0.470
10.2	YOLOv7-e6e	0.929	0.706	0.991	0.814

Table 5.7: Inference time for the best-performing models of each YOLOv7 version

Training	Model	Inference Time (ms)	
		GPU	CPU
11.1	YOLOv7-tiny	1.3	48.5
8.1	YOLOv7-e6	17.8	1659
10.2	YOLOv7-e6e	20.1	2007

²This is the best-performing YOLOv7-tiny model using k-folds cross-validation

³This is the best-performing YOLOv7 model before using batches 1, 2, and 3 as the training dataset

5.3 Detection of Waste, Graffiti, Traffic Signs, and Billboards

After collecting and censoring the images taken with a mobile device, the goal was to train a different model that would detect different objects found in public spaces while running on the server. These objects included waste containers, plastic bags, cardboard boxes, graffiti, traffic signs, and billboards. Also, to ensure that every person and license plate instance was detected and censored in the previous task, these two classes were still part of the training dataset.

Even though real-time detection is not a requirement for this task, the YOLOv7-e6e model will be used due to its previous success. Moreover, based on the results obtained in the "Detect Waste in Pomerania" project with EfficientDet, some of these models will also be used in the training process.

5.3.1 Datasets and Preprocessing

Considering the datasets discussed in chapter 4, such as TACO and Detect-waste+, were only focused on very specific waste objects and didn't have almost any instances for waste bags and cardboard boxes, different datasets had to be considered. As in the previous task, COCO and Open Images were used as the sources for the person and license plate classes, respectively. Similar to what was previously done, the approach taken for the training process of the remaining classes consisted of only selecting a few of those classes; finding the best dataset that includes them; training a model to detect those classes in batches 1, 2, and 3 images (mentioned in chapter 5.2.3); and use the annotated images as the new training dataset for the next iteration.

The images for the waste container and plastic bag classes were collected from Open Images, as well as the traffic sign and billboard classes. On the other hand, gathering cardboard images was the biggest challenge. Besides the lack of datasets with this specific class, the vast majority that was found only included indoor pictures of mostly boxes in good preservation conditions, which is the opposite of what is found on the streets. For this purpose, a few different options were evaluated, including the "corrugated carton" class from TACO (with less than 70 useful images) and even the instances found in the images collected in Madeira, which also combined for less than 100 images (from more than 16000 images). After an extensive search, a dataset from Amsterdam with around 1000 street-waste images was found in [56], which includes close to 1200 cardboard annotations, 1100 plastic bag annotations, and 1800 waste container annotations. Finally, a dataset named STORM (Safeguarding Cultural Heritage through Technical and Organisational Resources Management) with graffiti images from the streets of Greece was found in [57], consisting of more than 1000 images.

Similar to the previous task, FiftyOne was used to download images from Open Images and COCO, while Roboflow was the platform of choice to edit each dataset's annotations.



Figure 5.11: Example images from the datasets used for this task (in addition to the ones in figure 5.2)

5.3.2 Models

For this task, only the biggest YOLOv7 model - YOLOv7-e6e - was used since it achieved the best results for the previous task. As already mentioned, the state-of-the-art mAP@0.5 for this model is 0.744, considering an input size of 1280x1280. On the other hand, a few EfficientDet models were also trained, such as EfficientDet-D0, D3, and D4. At first, an attempt to use the code from the "Detect Waste in Pomerania" project was made. However, the code's architecture was already too adapted for their datasets' particularities and required training the model for detection (with EfficientDet) and classification (with its backbone, EfficientNet) in separate steps, which would require more time adapting it to this project's needs. For these reasons, it was decided that this project would rely on the repository their code was based on. Thus, the training code and model's weights were all obtained from [58].

Even though more recent versions are already available (the most recent is the EfficientDet-D7x with a mAP@0.5 of 0.743), the EfficientDet’s architecture is much more complex than the YOLOv7’s, which means a greater computational power is required to train such models. This is why the best model mentioned in the next section 5.3.3 is the EfficientDet-D4. The remaining versions were either too slow to train (considering the limited time available for this project) or wouldn’t even train with the minimum batch size of 1, considering the last dataset described in the same section. According to [4], the mAP@0.5 expected for each one of them is around 0.530, 0.659, and 0.684, respectively (as described in table 5.8).

Table 5.8: State-of-the-art accuracy results using YOLOv7 and EfficientDet models

Model	Input Size	mAP@0.50
YOLOv7-e6e	1280x1280	0.744
EfficientDet-D0	512x512	0.530
EfficientDet-D3	896x896	0.659
EfficientDet-D4	1024x1024	0.684
EfficientDet-D7x	1536x1536	0.743

5.3.3 Training Results

As in section 5.2.3, all the following training results were obtained with a RTX 3090 GPU using the best weights from training 10.2 in table 5.4 and the pre-trained weights from repository [58] (for EfficientDet). The input size chosen for each training also depended on the model, and it was the same as the corresponding input size mentioned in table 5.8. Most of the training sessions mentioned below used the hold-out cross-validation technique, with a ratio of around 70/30 (training/validation). However, the last models were also trained using the k-folds technique with five folds, as will be described in this section. Again, some of the images from Funchal were used as the testing set of these models. These images also have the corresponding people and license plates manually censored to ensure privacy.

The first training session was done with YOLOv7-e6e using the plastic bag and waste container classes from Open Images. Around 2000 images were downloaded for this purpose and then uploaded to Roboflow to edit the incorrect or irrelevant annotations. After editing, only 1258 images remained. These were used as the training set, achieving a mAP@0.5 and mAP@0.5:0.95 of 0.731 (0.575 for plastic bags and 0.886 for waste containers) and 0.499 after 50 epochs, respectively. This model was then used to detect these two classes in batch 1, 2, and 3 images. The three batches were again combined to form a single training dataset, and using the same weights as the previous model, a mAP@0.5 of 0.888 (0.851 for people, 0.930 for license plates, 0.907 for plastic bags, and 0.862 for waste containers) and a mAP@0.5:0.95 of 0.715 was achieved.

The same process was repeated for the traffic sign and billboard classes from Open Images and the graffiti class from STORM, which combined for 3022 images - 2000 of the first two classes and 1022 of the graffiti class. Using the same weights, this new training session achieved a mAP@0.5 and mAP@0.5:0.95 of 0.702 (0.764 for traffic signs, 0.632 for billboards, and 0.720 for graffiti)

and 0.492, respectively. The addition of these three new classes in batches 1, 2, and 3 images, combined for a total of seven classes in 2447 images. Using this new training dataset, the model achieved a mAP@0.5 of 0.770 (0.893 for people, 0.932 for license plates, 0.911 for plastic bags, 0.838 for waste containers, 0.881 for billboards, 0.932 for traffic signs, and 0.00 for graffiti) and a mAP@0.5:0.95 of 0.623.

The last dataset was very unbalanced regarding the ratio of annotations per class, particularly in the plastic bags, graffiti, and billboard classes. Overall, the validation set only had a total of 1548 annotations for seven classes, of which: 25.78% were people; 38.37% were license plates; 2.07% plastic bags; 11.76% waste containers; 1.49% billboards; 20.22% traffic signs; and 0.32% graffiti. Hence, to solve the small amount of plastic bags and waste container annotations, the previous 1258 images from Open Images were added to the training dataset. Regarding the graffiti class, all images from the STORM dataset were also added.

After detecting every missing class on the newly added images, the new dataset was composed of 4571 images with a total of 3292 annotations, of which: 21.29% (-4.48%) were people; 17.16% (-21.21%) were license plates; 6.56% (+4.49%) plastic bags; 21.63% (+9.87%) waste containers; 0.61% (-0.88%) billboards; 10.39% (-9.83%) traffic signs; and 22.36% (+22.03%) graffiti. A new training session was made using the weights from the previous training achieving a mAP@0.5 of 0.802 (0.858 for people, 0.933 for license plates, 0.519 for plastic bags, 0.860 for waste containers, 0.849 for billboards, 0.940 for traffic signs, and 0.656 for graffiti) and a mAP@0.5:0.95 of 0.644.

Table 5.9: Accuracy results using the YOLOv7-e6e model to detect 7 classes

Training	Dataset ⁴	Split (Train/Val)	mAP@0.50	mAP@0.50
1	Open Images (PB;WC)	881/377	0.731	0.499
2	Batches 1, 2, and 3 ⁵	1599/691	0.888	0.715
3	Open Images (B;TS) + STORM	2087/935	0.702	0.492
4	Batches 1, 2, and 3 ⁶	1648/799	0.770	0.623
5	Batches 1, 2, and 3 ⁷	3190/1398	0.802	0.644

Considering the YOLOv7-e6e training with the best results detecting seven classes - training 5 in table 5.9 - the validation set consisted of 1398 images with 701 annotations of people, 565 of license plates, 216 of plastic bags, 712 of waste containers, 20 of billboards, 342 of traffic signs, and 736 of graffiti. Running on these validation images, the inference time was around 21.7 ms with GPU - using a batch size of 8. In this case, figure 5.12 shows the plastic bag, waste container, and graffiti classes were the only ones with less than 90% correct predictions. Nevertheless, all of them correctly predicted in more than 70% of the time and were only confused with the background. The least correctly predicted class was the plastic bag, with 71%. On the other hand, the person and graffiti classes were the most confused with the background. In

⁴PB: represents the "Plastic bag" class; WC: represents the "Waste container" class; TS: represents the "Traffic sign" class; B: represents the "Billboard" class

⁵With 4 classes annotated

⁶With 7 classes annotated

⁷With the addition of Open Images (PB;WC) and STORM (with 7 classes annotated)

addition, some images from Madeira were used as the testing set of this model, and a few results can be seen in figure 5.13.

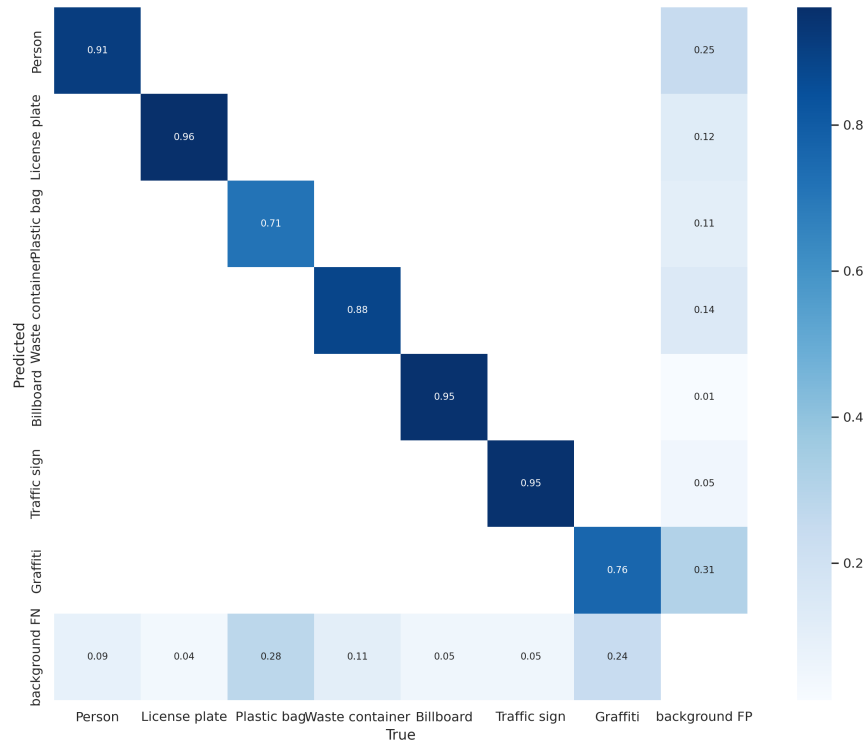


Figure 5.12: Confusion matrix of training 5 in table 5.9

The first approach to detect cardboard was to combine images of the "Corrugated carton" class from TACO - which was the only class from this dataset slightly similar to what was required - with images from batches 1, 2, and 3 that contained this class. However, each of these datasets only had around 70 images. Furthermore, after removing the irrelevant annotations, only 95 images were left. Hence, some data augmentation was applied to improve the dataset, particularly rotation (-15° and 15°) and exposure (-15% and 15%) - which were the only modifications that seemed fit for this use-case. Still, the result was a dataset with only 285 images. Using the weights from training 5 in table 5.9, this dataset achieved a mAP@0.5 and mAP@0.5:0.95 of 0.980 and 0.899, respectively. Yet, after testing the model on different images from Madeira, it became clear that these values were only a result of overfitting and didn't reflect the actual accuracy.



Figure 5.13: Testing images annotated using training 5 in table 5.9 (white bounding boxes for "plastic bag", green for "graffiti", brown for "waste container", grey for "traffic sign, and purple for "license plate")

As mentioned in section 5.3.1, the best dataset found for this purpose [56] included a total of around 1000 images with cardboard, plastic bag, and waste container annotations. So, the next approach was to use these cardboard annotations to train the model and then use it to annotate the images from the training 5 (table 5.9) dataset. After correcting these annotations in Roboflow, there were a total of 462 images with cardboard annotations. Relying on the weights from training 5, the model achieved a $mAP@0.5$ and $mAP@0.5:0.95$ of 0.641 and 0.344, respectively. Since the images from the Amsterdam dataset had a background so similar to what was found in Madeira's pictures, all of them were added to the previous dataset after being annotated with the remaining seven classes.

The final dataset consisted of 5588 images with eight different classes after editing. Training with the same weights as mentioned above, a $mAP@0.5$ of 0.782 (0.840 for people, 0.848 for license plates, 0.655 for plastic bags, 0.879 for waste containers, 0.819 for billboards, 0.951 for traffic signs, 0.702 for graffiti, and 0.563 for cardboard) and a $mAP@0.5:0.95$ of 0.605 was achieved. After 50 more epochs the results didn't improve, only achieving a $mAP@0.5$ of 0.775

(0.835 for people, 0.854 for license plates, 0.655 for plastic bags, 0.872 for waste containers, 0.764 for billboards, 0.951 for traffic signs, 0.698 for graffiti, and 0.569 for cardboard) and a mAP@0.5:0.95 of 0.596.

Considering the overall results already achieved with the final dataset, the last training was repeated with k-folds cross-validation instead of hold-out. After splitting, each fold had 1117 images with 14.65% person, 13.61% license plate, 11.59% plastic bags, 26.08% waste containers, 0.44% billboards, 7.51% traffic signs, 16.66% graffiti, and 9.46% cardboard annotations. To keep the training time below 10 hours, each fold was initially trained for only 10 epochs - meaning 50 epochs in total. On average, the overall mAP@0.5 and mAP@0.5:0.95 for each fold was 0.781 (0.879 for people, 0.833 for license plates, 0.723 for plastic bags, 0.885 for waste containers, 0.803 for billboards, 0.929 for traffic signs, 0.730 for graffiti, and 0.471 for cardboard) and 0.611, respectively. Finally, the previous model was retrained for 10 more epochs (for each fold) to see if these values would improve. However, that was not the case. This time, the overall mAP@0.5 and mAP@0.5:0.95 for each fold was 0.781 (0.881 for people, 0.829 for license plates, 0.716 for plastic bags, 0.876 for waste containers, 0.803 for billboards, 0.925 for traffic signs, 0.746 for graffiti, and 0.473 for cardboard) and 0.610, respectively.

Table 5.10: Accuracy results using the YOLOv7-e6e model to detect 8 classes

Training	Dataset ⁸	Split (Train/Val)	mAP@0.50	mAP@0.50	Cross-validation
6	Amsterdam (C)	338/124	0.641	0.344	Hold-out (70/30)
7	8 classes ⁹	3908/1677	0.782	0.605	Hold-out (70/30)
7.1	8 classes	3908/1677	0.775	0.596	Hold-out (70/30)
8	8 classes	4468/1117	0.781	0.611	K-folds (k=5)
8.1	8 classes	4468/1117	0.781	0.610	K-folds (k=5)

Considering the YOLOv7-e6e training with the best results detecting eight classes - training 7 in table 5.10 - the validation set consisted of 1636 images with 748 annotations of people, 648 of license plates, 471 of plastic bags, 1274 of waste containers, 21 of billboards, 325 of traffic signs, 804 of graffiti and 420 of cardboard. Running on these validation images, the inference time was around 23.7 ms with GPU - using a batch size of 4. In this case, figure 5.14 shows the plastic bag, graffiti, and cardboard classes were the only ones with less than 88% correct predictions. As expected, the newly added cardboard class was the least correctly predicted, with only 61%. Finally, some images from Madeira were used as the testing set of this model, and a few results can be seen in figure 5.15.

⁸C: represents the "Cardboard" class

⁹The combination of batch 1, 2, and 3 annotated images with Open Images (PB;WC), STORM, and Amsterdam datasets

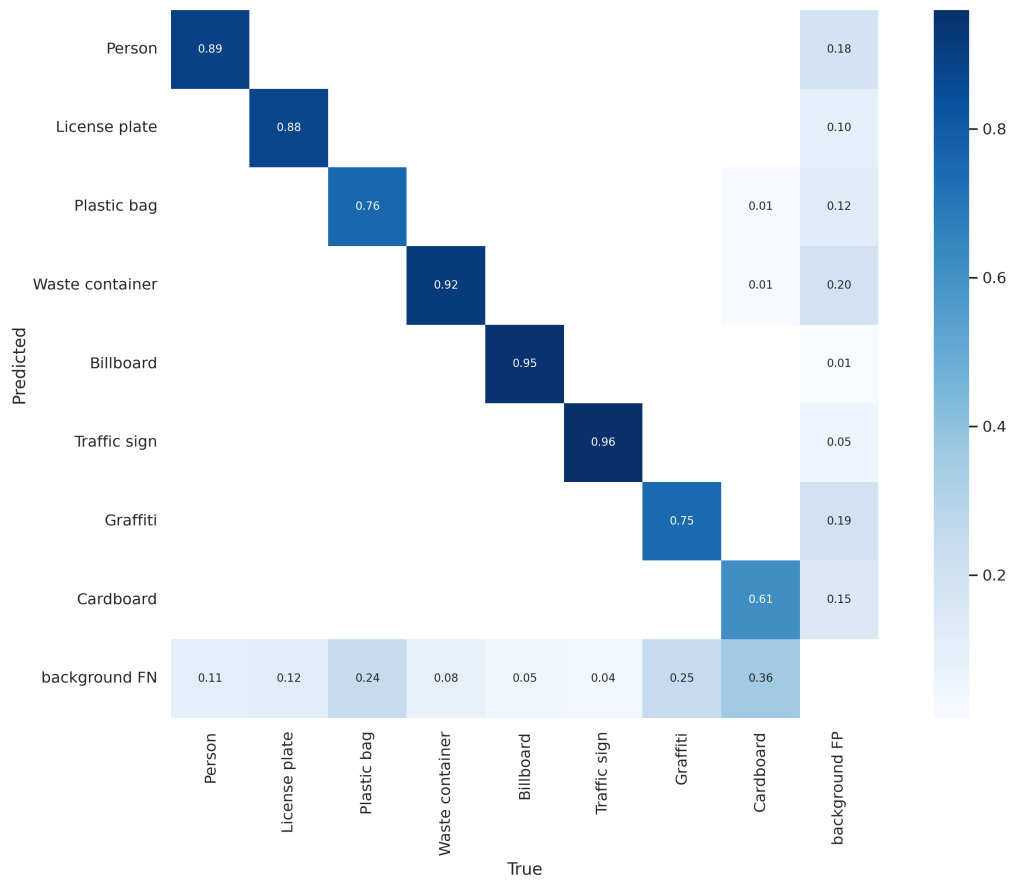


Figure 5.14: Confusion matrix of training 7 in table 5.10

Relying only on the last dataset with eight different classes across 5588 images, new training sessions were made using EfficientDet models, as mentioned in section 5.3.2. The first to be trained was the EfficientDet-D0, using a batch size of 32. After 100 epochs, the $mAP@0.5$ and $mAP@0.5:0.95$ were only 0.336 and 0.204, respectively. An EfficientDet-D3 was then trained using a batch size of 4, which was the maximum size supported by the GPU. After 100 epochs, the $mAP@0.5$ and $mAP@0.5:0.95$ still remained very low, with only 0.451 and 0.297, respectively. Lastly, an EfficientDet-D4 was also trained (the batch size had to be reduced to 2). In comparison to the previous training, the $mAP@0.5$ only improved by 0.7%, while the $mAP@0.5:0.95$ improved by 1.3%.

All previous EfficientDet models were trained with a minimum learning rate of $1e-4$, and there was a significant difference between train and test loss - between 10 and 20% - during their training. For example, while training EfficientDet-D3, the best train loss was around 0.42, while the test loss was 0.57. A reason for this overfitting could be a learning rate too high. However, after several tries with maximum learning rates below $1e-4$, the results remained very poor. It was concluded the problem was related to the imbalance in the training and validation set classes. Hence, a new training session with EfficientDet-D4 was done using the same process of k-folds



Figure 5.15: Testing images annotated using training 7 in table 5.10 (blue bounding boxes for "plastic bag", light green for "graffiti", pink for "cardboard", dark green for "waste container", white for "traffic sign, and orange for "license plate")

cross-validation as previously, which split the data while balancing the classes. After 100 epochs, the overall $mAP@0.5$ and $mAP@0.5:0.95$ were 0.689 (0.746 for people, 0.784 for license plates, 0.750 for plastic bags, 0.895 for waste containers, 0.162 for billboards, 0.878 for traffic signs, 0.724 for graffiti, and 0.586 for cardboard) and 0.484, respectively. During this training, despite the performance improvement, the test loss appeared to decrease faster than the train loss (which could be a sign of underfitting). Therefore, the last model was retrained with the same weights for 50 additional epochs per fold, using a minimum learning rate of $1e-3$. The resulting $mAP@0.5$ and $mAP@0.5:0.95$ were 0.802 (0.799 for people, 0.833 for license plates, 0.787 for plastic bags, 0.901 for waste containers, 0.844 for billboards, 0.893 for traffic signs, 0.734 for graffiti, and 0.628 for cardboard) and 0.569, respectively. Finally, some images from Madeira were used as the testing set of this model, and a few results can be seen in figure 5.17.

¹⁰In training 9 and 10, EfficientDet-D0 and D3 models were used, respectively. In the remaining, only EfficientDet-D4 was used.

Table 5.11: Accuracy results using EfficientDet models to detect eight classes

Training ¹⁰	Dataset	Split (Train/Val)	mAP@0.50	mAP@0.50	Cross-validation
9	8 classes	3908/1677	0.336	0.204	Hold-out (70/30)
10	8 classes	3908/1677	0.451	0.297	Hold-out (70/30)
11	8 classes	3908/1677	0.458	0.310	Hold-out (70/30)
12	8 classes	4468/1117	0.689	0.484	K-folds (k=5)
12.1	8 classes	4468/1117	0.802	0.569	K-folds (k=5)

Considering the EfficientDet training with the best results detecting eight classes - training 12.1 in table 5.11 - the validation set consisted of one fold with 1117 images (its class balance was already mentioned above). Running on these validation images, the inference time was around 37.5 ms with GPU - using a batch size of 2. In this case, figure 5.16 shows the plastic bag, graffiti, and cardboard classes were the only ones with less than 82% correct predictions. Moreover, the newly added cardboard class and the billboard class were the least correctly predicted, with only 69% and 45%, respectively. Finally, some images from Madeira were used as the testing set of this model, and a few results can be seen in figure 5.17.

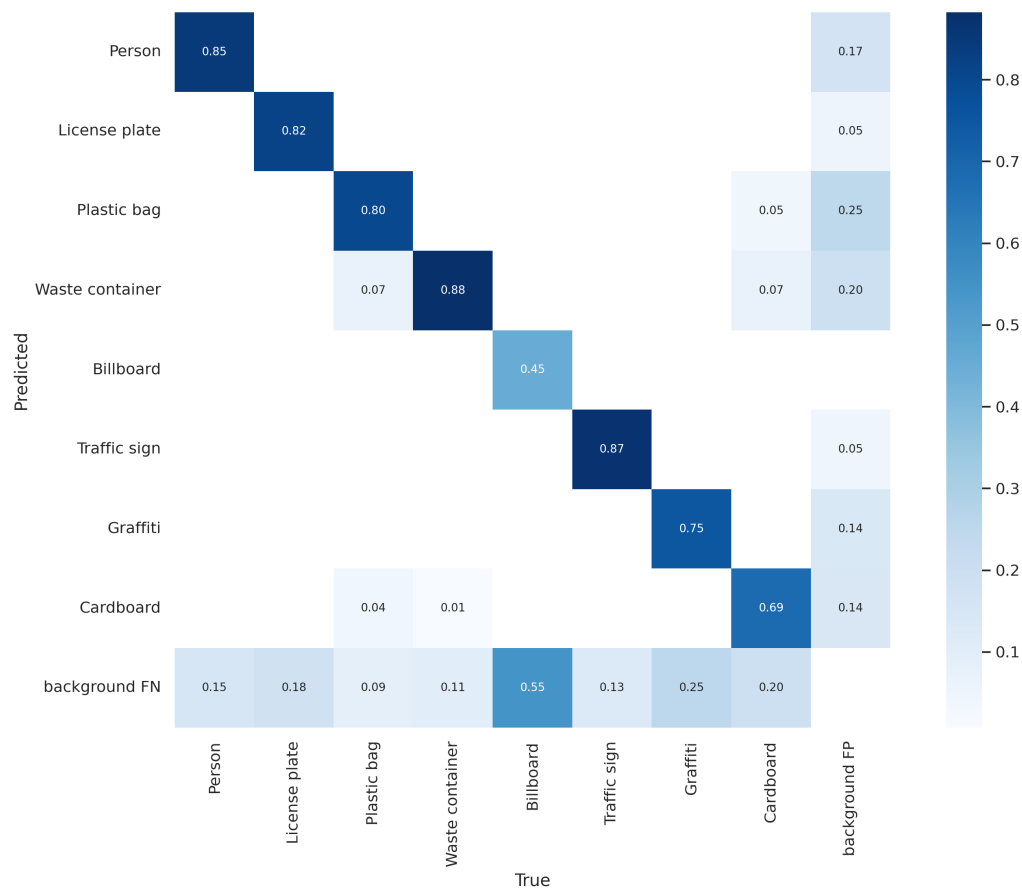


Figure 5.16: Confusion matrix of training 12.1 in table 5.11



Figure 5.17: Testing images annotated using training 12.1 in table 5.11 (blue bounding boxes for "plastic bag", purple for "cardboard", red for "waste container", yellow for "person", green for "traffic sign", and blue for "license plate")

5.3.4 Summary

A combination of five different datasets was used to mainly train two different models - YOLOv7-e6e and EfficientDet-D4 - to detect eight different classes: people, license plates, plastic bags, waste containers, billboards, traffic signs, graffiti, and cardboard. The main challenges for this task were finding the right dataset for each class and balancing the number of annotations per class to prevent over or underfitting. For this reason, it was used not only hold-out cross-validation but also k-folds cross-validation. The most relevant values for the best-performing models are displayed in table 5.12.

Table 5.12: Accuracy results for the best-performing models - second task

Training	Model	N. classes	mAP@0.50	mAP@0.50	Cross-validation
5	YOLOv7-e6e	7	0.802	0.644	Hold-out (70/30)
7	YOLOv7-e6e	8	0.781	0.610	Hold-out (70/30)
12.1	EfficientDet-D4	8	0.802	0.569	K-folds (k=5)

Chapter 6

Discussion

This chapter is divided into three main sections - "Datasets", "Accuracy", and "Real-time Detection and Training Time". In the first one, every dataset used in chapter 5 will be analyzed and compared to the state-of-the-art mentioned in chapter 3. The "Accuracy" section is divided into two different subsections - "Detection of People and License Plates" and "Detection of Waste, Graffiti, Traffic Signs, and Billboards". Here, the most relevant results achieved for each task in the previous chapter will be discussed and explained. Lastly, in "Real-time detection and training time" the inference and training time of each model used for each task will be analyzed.

6.1 Datasets

As mentioned in chapter 5, the datasets related to waste detection that were discussed in the literature review - such as TACO and detect-waste+ - didn't cover the majority of classes this project aimed to detect. Moreover, the only common classes represented in these datasets - plastic bags and cardboard - were very poorly represented and didn't share many characteristics with the examples found in Madeira's streets (and in Portuguese streets in general) - such as their shapes and colors. Thus, several thousand pictures were taken in the streets of Funchal, resulting in an accurate dataset of its public landscape. On the other hand, these images were not annotated, and doing it manually would take a large amount of time, so a different approach was initially taken.

In the detecting people and license plates task, the first source for person annotations was the Open Images dataset. However, after manually analyzing each of them in Roboflow, many incorrect annotations (at least for this use case) were found - including people representations in cartoons, billboards, paintings, toys, etc. Therefore, MS COCO ended up being the source for this class since it is a more well-known and widespread benchmark dataset, and it didn't contain as many incorrect annotations as the previous one. On the other hand, Open Images provided good-quality annotations for license plates. This class has the advantage of being very homogeneous since the majority of license plates follow a standard shape, particularly in most European countries.

Open Images also provided good-quality annotations for other classes, such as waste containers, plastic bags, and traffic signs. In the second task, waste containers were the first detected class. The images from Open Images were very diverse and included waste containers in many different shapes and colors. However, the trained models were not as accurate in detecting some typical waste containers from Madeira - with lighter colors and made from steel - as in detecting standard black containers. This was corrected after training with images from Madeira, where these examples could be found. On the other hand, the majority of plastic bag images collected were neither waste plastic bags nor on the streets, which is the desired background for this project. This was compensated with around 1100 annotations from the "Amsterdam" dataset.

Similar to the license plate annotations, good-quality traffic sign images were easy to find in Open Images since they also share many visual similarities worldwide. On the other hand, the billboard images found in this dataset were very heterogenous and deviated too much from the types of billboard targeted for this project - mainly big advertising billboards on sidewalks (or close to the street level) and smaller ones on bus stops. Since this was not one of the most relevant classes for this project and the time to manually compile a more accurate dataset was limited, the only annotations used as training data for this class were a compilation of the ones found in Madeira's dataset, which only combined for 58 images (with 62 annotations in total) out of more than 16000 images. Being such a small compilation, the detection improvement of this class was postponed for future work.

After searching for public datasets to detect the seventh class - graffiti - the STORM dataset was the only one found with a very diverse and large amount of images with these particular annotations. In addition, this dataset consisted of pictures taken on Greek streets, which have some similarities with Portuguese streets.

Finally, the last class to be detected - cardboard - was the most challenging to find. Unlike the examples found in datasets such as TACO and detect-waste+, which include very diverse backgrounds and are more focused on smaller-sized pieces of waste, the goal of this project was to detect cardboard boxes with different sizes and colors and in various states of conservation - either brand new or torn into smaller pieces, for example. The only dataset that was found to fit these requirements was the "Amsterdam" dataset, whose images aligned with the street-level view required. As already mentioned, besides including around 1200 annotations of cardboard, this dataset also contained more than 1000 annotations of waste bags, which helped improve waste bag detection.

Overall, even though the abovementioned datasets had a very significant impact on the training process of this project, the best results were obtained after using images from Madeira's dataset as part of the training set, as will be explained in the next section 6.2. The fact that the testing datasets only included images of Madeira's public landscape, where certain objects and types of waste can differ from other places, should be related to this performance improvement since the training data is accurately representing reality.

6.2 Accuracy

6.2.1 Detection of People and License Plates

Given the real-time detection requirement for the people and license plate detection task, the first model to be trained was the YOLOv7-tiny - the least complex YOLOv7 model. Relying on images from COCO and Open Images, the best accuracy results initially achieved by this model were a mAP@0.5 of 0.693 and a mAP@0.5:0.95 of 0.401 (training 4 in table 5.2). As mentioned in section 5.2.3, the YOLOv7-tiny model was trained using an input size of 640x640 instead of the 320x320 or 416x416 standard sizes cited in [24], mainly because every image collected in Madeira was 1280x1920 and 640x640 could be an intermediate size that would not downgrade the image's quality too much.

Even though these results were not as high as expected for this project's goals and a standard input size was not used, there is a substantial improvement from a mAP@0.5 of 0.528 (for a YOLOv7-tiny with an input size of 416x416) to 0.693 (with an input size of 640x640), in comparison to the state-of-the-art values in table 5.1. It is also worth mentioning the accuracy improvements seen between training 1 and 2 - which achieved a mAP@0.5 of 0.431 and 0.519, respectively - and training 3 and 4. In training 3, besides using people annotations from COCO instead of Open Images - which have better quality, as discussed in section 6.1 - the training dataset was smaller and only had one class - person - in comparison to training 2. With these adjustments, there was a mAP@0.5 improvement of 22.4% by the people class between training 2 and 3. Even though the better quality of COCO's annotations may have been an important factor in this improvement, training with a single class and with a smaller dataset must have had an impact as well. The large dimension of the previous dataset (with 10000+ images) might have introduced too diverse annotations for the same class, being less efficient to train such a small model with only 50 epochs.

Considering the accuracy per class, there was a significant difference between the mAP@0.5 of 0.607 for people and 0.780 for license plates, regardless of the larger amount of person annotations in the validation set (4296 annotations against 1512). This outcome can be explained by the homogeneity in license plate annotations since the majority follows a standard visual appearance. Moreover, the European Union standard license plates are well-represented in the images from Open Images. On the other hand, different people have very different visual characteristics, such as skin color, weight, and types of clothes. Consequently, it is more challenging for a computer vision model to detect patterns in classes like this one.

Since the results obtained with YOLOv7-tiny were not as good as expected, more complex models were trained to compare results between them and choose which one could be the best for this task. Even though YOLOv7-e6 and YOLOv7-e6e were not optimized for real-time detection like the first model, they could be used in future iterations of this project if the edge device's computational power is improved with a GPU, for example.

During the training process of YOLOv7-e6, after analyzing the datasets and corresponding results, it was realized that every image with license plate annotations also included people that were not annotated. Hence, a model was trained only to detect people and then used to annotate

these images. Before this correction, the best accuracy was achieved in training 6 (table 5.3) with an overall mAP@0.5 of 0.757, including a mAP@0.5 of only 0.689 for people. On the other hand, after adding the missing annotations, the last training achieved an overall mAP@0.5 of 0.797, including a mAP@0.5 of 0.761 for people (an improvement of around 7.2%).

The overall detection improvement between the best training with YOLOv7-tiny and YOLOv7-e6 can be noticed by comparing the example images in figures 5.4 and 5.6. Unlike the first one, in figure 5.6, every license plate and person present in those images were correctly detected and with higher confidence. However, it can also be seen some confusion between the car's side mirror and a person. This might be related to the fact that some images from COCO have annotations of people that are very far away in the background and consequently can be confused with a "blurred" object such as this one. In addition, the side mirror has a rounded shape, which can possibly be mistaken for a human head moving fast from the camera's perspective. Nevertheless, both training 8 and 8.1 revealed a higher mAP@0.5 compared to the state-of-the-art accuracy of 0.735, mentioned in table 5.1.

After training the YOLOv7-e6e model with the same dataset as in training 8.1 (table 5.3) and testing it with some images from Madeira, there were some incorrect detections, such as detecting a group of people as one single detection, detecting specific body parts as a person, or detecting people that are very far away and can't be recognized. Hence, these annotations were corrected in Roboflow, and the training dataset was updated. However, using this updated dataset, the mAP@0.5 for the person class decreased from 0.751 (training 9 in table 5.4) to 0.733 (training 9.2). This correction process was done manually in thousands of images, so there is a high probability that many incorrect annotations were not filtered, particularly annotations of people very far away and barely recognizable. This might be one of the reasons why the accuracy decreased since the model might still be detecting the same instances that are now considered False Positives.

In order to improve the model's accuracy even further, YOLOv7-e6e was retrained with a new training dataset consisting only of batches of annotated images from Madeira. After training with the first one, the model was used to annotate the second batch, which was then added to the training dataset. This process was repeated once again, resulting in a dataset of 2201 images that achieved a mAP@0.5 of 0.960 (0.929 for people and 0.991 for license plates). In comparison to the highest accuracy achieved with the previous dataset, in training 9, there was an overall mAP@0.5 improvement of 16.7%. This difference is even larger - around 21.6% - when compared to the state-of-the-art accuracy for this model, 0.744. Some improvements can also be seen in figure 5.8, where all annotations were correctly detected with higher confidence. Furthermore, the car's side mirror was no longer mistaken for a person, and a third license plate was detected in the first image, which was too far away (and consequently too small) to be detected by the previous models.

In conclusion, using images from Madeira as the training set had a positive impact on the model's performance. The fact that these images shared similar backgrounds, colors, and lighting conditions with the public landscape where the model would be trained on was the main reason

behind the accuracy improvements. A comparison between the results of each YOLOv7 model can be seen in figure 6.1.



Figure 6.1: A compilation of testing images obtained with three different YOLOv7 models

Since the dataset consisting of batches 1, 2, and 3 was the one that led to the best results achieved by any YOLOv7 model, the YOLOv7-tiny model was retrained with this dataset. As expected, there were significant accuracy improvements when compared to training 4 (in table 5.2). The highest accuracy was achieved in training 11.1 with a $mAP@0.5$ of 0.813 (0.802 for people and 0.825 for license plates) and a $mAP@0.5:0.95$ of 0.551. This represents an overall improvement of 12% and 15%, respectively. Regarding each class, the $mAP@0.5$ improved by

19.5% for people and 4.5% for license plates. The improvement is noticeable when comparing figure 5.4 and 5.10, particularly in the third image of figure 5.10, where every person was correctly detected.

The same model was also trained with k-folds cross-validation instead of hold-out to see how much the data balancing would affect the results. Splitting the dataset into five folds of 441 images - each with around 38.33% person and 61.67% license plate annotations - the accuracy values achieved in training 12 were significantly below the ones in training 11 (both achieved after 50 epochs). After 50 more epochs per fold, the mAP@0.5 and mAP@0.5:0.95 were still 28.0% and 19.7% lower than in training 11, and 7.6% and 9.1% lower than in training 11.1, using the same amount of epochs. On one hand, the model used in training 11 and 12 was consecutively trained using the hold-out technique before this experiment was done, so this might have influenced the results achieved in training 12. On the other hand, the fact that training with the same dataset had such different outcomes after balancing its data (when using k-folds validation) may indicate that in previous training iterations, there was some overfitting inflating the accuracy values. A comparison between the results achieved with the different techniques is displayed in figure 6.2.



Figure 6.2: Comparison between the testing images achieved in training 4 and 11.1 (in tables 5.2 and 5.5)

6.2.2 Detection of Waste, Graffiti, Traffic Signs, and Billboards

For the second task mentioned in chapter 5, two different models' architectures were trained - YOLOv7 and EfficientDet. Since these models didn't have a real-time detection requirement as in the previous task, the goal was to use more complex models and consequently achieve better accuracy for a larger variety of classes. Considering the good results previously achieved with the YOLOv7-e6e model, this was the first one to be trained to detect eight different classes - people, license plates, waste containers, waste plastic bags, cardboard, traffic signs, billboards, and graffiti. The first two classes are shared between the two tasks to ensure a more robust privacy filter in case the model running in real-time doesn't censor every expected instance of these classes. Therefore, the first training made with YOLOv7-e6e in this task relied on the weights from training 10.2 (in table 5.4).

The first approach used in this training process was to train the YOLOv7-e6e model with two or three classes simultaneously and use it to annotate around 3000 images from Madeira (batches 1, 2, and 3). Afterwards, these images were manually analyzed in Roboflow and then used as the training set of the previous model. This method was revealed to be very time-efficient since smaller datasets meant less training time. Otherwise, by starting with a very large dataset that included every required class, the extended training time would postpone its analysis and corresponding improvements for several hours.

The first two classes to be trained were the waste container and plastic bag. After training with batches 1, 2, and 3 already annotated, the overall mAP@0.5 was 0.888. This iteration already had high accuracy values for the newly added classes - with a mAP@0.5 of 0.862 and 0.907, respectively. However, after testing this model on some testing images from Madeira, these results were revealed to have been affected by overfitting to the training data. The main reason for this overfitting was the fact that only 2.07% of the validation set was composed of images with plastic bag annotations, as mentioned in section 5.3.3. And regarding the waste container class, even though it had a higher representativity (around 11.76%), the presence of different types of waste containers in Madeira required more data to have the same accuracy across every one of them.

The following training session's goal was to detect traffic signs, billboards, and graffiti. After adding these new classes to the previous training dataset, the overall mAP@0.5 decreased to 0.770, including a mAP@0.5 of 0.932, 0.881, and 0.00, for each new class. Similar to the license plate class, the standard shape of traffic signs and the homogeneity between their images led to a higher accuracy than every other class, as expected. On the other hand, despite the high accuracy value achieved by the billboard class, there was significant overfitting to training images as in the plastic bag class, particularly because from the 1548 images that composed the validation set, only 23 of them (1.49%) included billboard annotations. The same issue affected the graffiti class since it was the most poorly represented in the validation set - with only 5 images in total (0.32%).

To solve this imbalance across classes and improve the model's performance, images including waste containers, plastic bags, and graffiti annotations (from Open Images and STORM) were added to the dataset. It is worth mentioning that every added image was annotated with every

missing class when needed. Altogether there was a representativity improvement of 9.87%, 4.49%, and 22.03%, respectively. This correction was reflected in the mAP@0.5 achieved by each class after training the model on the updated dataset. The waste container and graffiti classes improved to 0.860 (+2.2%) and 0.656 (+65.6%), respectively. On the other hand, the plastic bag class decreased to 0.519 (-39.2%) due to the addition of a wider variety of annotations from Open Images, which actually didn't fit the reality found in Madeira's public landscape (as mentioned in section 6.1).

In figure 6.3, an improvement in the plastic bag class can be seen despite the significant decrease in accuracy, which confirms the previous overfitting. Moreover, in the third image of training 4, an example of a FP detection of the waste container class - which didn't occur in training 5 - is also displayed. Nevertheless, the overall results confirmed the importance of having a well-balanced dataset, particularly one with such heterogeneous classes.



Figure 6.3: Comparison between the testing images achieved in training 4 and 5 (in table 5.9)

Among the datasets mentioned in chapter 4, the only one including cardboard annotations that fit this project's requirements was TACO, in particular in its "Corrugated carton" class. However, only around 70 images were found to be relevant, which was too limited considering the variety of shapes and colors cardboard waste could have. Even by combining them with the also small amount of images from Madeira's dataset and using data augmentation, the results were still very poor due to overfitting. On the other hand, adding the images from Amsterdam's dataset to the

dataset used in training 5 (table 5.9) led to a mAP@0.5 of 0.563 for cardboard and some accuracy improvements in other classes, particularly waste containers and plastic bags - with 0.879 (+1.5%) and 0.686 (+13.6%), respectively. This improvement reflected the presence of more than 1000 annotations for each of these classes in the new images from Amsterdam, which aligned with the reality seen in a street landscape as in Madeira. Furthermore, significant improvement was also seen in the graffiti class - resulting in a mAP@0.5 of 0.702 (+4.6%). The fact that Amsterdam's images also showed some graffiti on walls and waste containers must have been a reason why.

Even though this model was already able to detect cardboard, the mAP@0.5 for this class was still relatively low in comparison to the other classes that had an average mAP@0.5 of 0.813, mainly due to being such a heterogeneous class of objects. Moreover, in its corresponding confusion matrix (in figure 5.14), a slight confusion can be seen between plastic bags and waste container (1% in both) predictions and cardboard ground-truth. This should be related to the fact that many cardboard annotations in Amsterdam's dataset are surrounded by waste container and waste bag annotations, which sometimes are inevitably overlapped.

Nevertheless, the same model was retrained for 50 more epochs to see if some accuracy improvements could be achieved, and the overall mAP@0.5 decreased by 0.7% to 0.775 (in training 7.1). The imbalance across classes might have had an influence on this outcome. Thus, training 7 was repeated using k-folds cross-validation to confirm if it was true. After only 10 epochs per fold, the average results - a 0.1% smaller mAP@0.5 than in training 7 and a 0.6% bigger mAP@0.5:0.95 - were already very similar to the ones achieved with the hold-out technique, confirming the class imbalance on the previous training dataset. On the other hand, training each fold for another 10 epochs (training 8.1) didn't improve the previous results. This outcome might indicate there is a need to improve the dataset by adding more images from Madeira - with more annotations for the most heterogeneous classes with less representativity, such as billboards, cardboard, and plastic bags.

As discussed in chapter 4, the positive results achieved in the "Detect Waste in Pomerania" project with EfficientDet models, particularly EfficientDet-D2, could suggest that this model architecture would be a good fit for a similar use-case of detecting objects on the streets. Therefore, after the several abovementioned experiments done with YOLOv7 models, the eight classes dataset from training 7 and 7.1 was used as the training dataset for some EfficientDet models. An EfficientDet-D0, D3, and D4 were trained for 100 epochs with the same dataset, and the highest mAP@0.5 and mAP@0.5:0.95 achieved were 0.458 and 0.310 (in training 11) - a 22.6% smaller mAP@0.5 than the state-of-the-art value mentioned in table 5.8. The difference between train and test loss during the last training was significant (around 10 and 20%), which led to the conclusion that the model was overfitting the training data even more than previously seen with YOLOv7 models. After trying different maximum learning rate values, the accuracy results remained unchanged, indicating that the class imbalance previously seen could be one of the explanations for the lack of performance.

Hence, the last model (EfficientDet-D4) was retrained using the same k-folds technique as with YOLOv7-e6e to balance the training data across every class. This approach revealed significant

improvements after 100 epochs per fold, achieving an average mAP@0.5 and mAP@0.5:0.95 of 0.689 (0.746 for people, 0.784 for license plates, 0.750 for plastic bags, 0.895 for waste containers, 0.162 for billboards, 0.878 for traffic signs, 0.724 for graffiti, and 0.586 for cardboard) and 0.484, respectively. Compared to training 11, the mAP@0.5 and mAP@0.5:0.95 values increased by 23.1% and 17.4%. However, the mAP@0.5 still just surpassed the state-of-the-art accuracy by 0.4%.

During the training process of the last model, the faster decline of test loss compared to train loss indicated some underfitting. Thus, the same model was retrained with a minimum learning rate of $1e-3$ (instead of $1e-4$) for 50 more epochs per fold, further improving the mAP@0.5 and mAP@0.5:0.95 to 0.802 and 0.569, respectively. Regarding the mAP@0.5 achieved per class, the most relevant improvements were in the person (+5.3%), plastic bag (+3.7%), graffiti (+1.0%), and cardboard (+4.2%) classes. Actually, the biggest improvement was seen in the billboard class (+68.2), but such a significant improvement with such a small amount of annotations was just the result of the previously discussed overfitting. This was confirmed by the small percentage of correct detections for this class in the corresponding confusion matrix (figure 5.16) - around 45%. The same confusion described in the YOLOv7-e6e experiments between waste containers, plastic bags, and cardboard can also be seen here.

Considering the performance achieved with EfficientDet-D4, there are some important aspects to compare with the YOLOv7-e6e's performance. On one hand, the highest mAP@0.5 achieved by EfficientDet-D4 using the hold-out technique was 0.458 - 31.7% lower than the one achieved with YOLOv7-e6e. Since EfficientDet models are much more complex than YOLOv7's, the fact that the dataset was randomly split and consequently unbalanced might have been the main factor contributing to this difference. On the other hand, when trained using k-folds cross-validation, EfficientDet-D4 achieved a 2.1% higher mAP@0.5 in training 12.1 than YOLOv7-e6e in training 8.1. However, there are significant performance differences when comparing the annotations in figures 5.15 and 5.17. In figure 5.17, the majority of detections have much lower confidence; the graffiti on the first image is not detected; one of the waste containers is mistaken for cardboard, and a motorcycle is mistaken for a person on the second image; and on the last image, none of the plastic bags were detected.

Even though the model used in training 12.1 was trained for 150 epochs instead of 100, this still should be a fair comparison because training 8.1 was the result of several consecutive training iterations of the same model since training 10.2 of the first task, while training 12.1 of the second task resulted of only two consecutive training iterations of the model in training 11. Furthermore, the standard input size of EfficientDet-D4 is 1024 instead of 1280, as in YOLOv7-e6e, which should make object recognition in an image more difficult because of the lack of resolution.

In conclusion, despite EfficientDet-D4 not achieving the same performance as YOLOv7-e6e with images from Madeira, more complex and accurate EfficientDet models with a larger input size should be superior to YOLOv7 models for this use case, considering a class-balanced dataset.

6.3 Real-time Detection and Training Time

Unfortunately, even though the ultimate goal of the first task was to test the model on an Android mobile device, this was not fulfilled. To run the model on such a device, it was required to convert the YOLOv7 model to a "tflite" (tensorflow lite) format. However, the code available in [54] to perform this conversion creates a "tflite" file with a different input tensor format than the one Android Studio expects, which makes it incompatible. In previous versions of YOLOv7, such as YOLOv5, this incompatibility didn't exist. This version and the most recent one - YOLOv8 - both were developed by the same author, "Ultralytics" [28, 25], and both are compatible with Android [59], which leads to the conclusion that the author of YOLOv7 must have made changes to the previous code that led to this problem. Using YOLOv8 models instead would have been the best solution, but unfortunately, this issue was detected too late to retrain the models with our datasets due to poor planning.

Nevertheless, considering the YOLOv7-tiny model that achieved the best accuracy results (training 11.1), the corresponding inference time was around 1.3 ms with a RTX 3090 GPU and 48.5 ms with an 11th Gen Intel i9-1199F CPU. In other words, if the mobile device used to detect people and license plates had a similar CPU, it would be able to detect these classes at around 20 FPS. For this use-case, around 3 to 5 FPS should be a good enough detection rate considering that these pictures will be taken in a city environment where the landscape doesn't change every second and also the vehicle where the mobile device will be placed is driving at a limited speed. Thus, if the abovementioned CPU was used, it would have a detection speed four times larger than required, for example. Since the person and license plate classes should not require significant changes to its training data in the foreseeable future, the amount of time it takes to train the model for this task is not as relevant as in the second task since the need to retrain it will not be as high. Nevertheless, the YOLOv7-tiny model in training 11.1 took less than 2 hours to train for 50 epochs (on GPU).

On the other hand, the YOLOv7-e6e and EfficientDet-D4 models that achieved the best accuracy for eight different classes (training 7 and 12.1) had an inference time of around 23.7 ms (42 FPS) and 37.5 ms (27 FPS) on GPU. In comparison to the YOLOv7-e6e model detecting two classes and using a dataset of 2201 images (in training 10.2), this model was around 3.6 ms slower due to the larger and more complex training dataset with 5585 images.

There is no real-time detection requirement for this task since this data needs to be reviewed by a human worker, and regardless of the outcome, it will certainly not affect the company's schedule for the next couple of days. Nevertheless, it is important to know how much time it will take to analyze a full workday of frames to choose the model that better fits, for example. As mentioned above, these classes will need more training data and training iterations to improve their detection, which means the training time and required computational power needed for each model are important aspects to consider for the system's architecture. In this case, the YOLOv7-e6e model of training 7 took more than 8 hours to complete, while the EfficientDet-D4 of training 12 took around 20 hours (around 2.5 times slower). The selection of the best EfficientDet model

to train during this experiment was also limited between versions D0 and D4 due to the extended training time it required. As an example, if training 12 had been done with EfficientDet-D5, the training time would increase to more than 30 hours.

Chapter 7

Conclusions and Future Work

This chapter highlights the conclusions drawn from all the experiments and corresponding results mentioned in previous chapters. In the first section, a summary of the achieved goals for this project is laid out. A critical analysis follows, pointing out what was successful during this project's development and what will need further improvement. Finally, some topics that were found to be promising moving forward will be presented as future work.

7.1 Main Contributions

Through the course of this work, it became clear that there is a very small amount of research regarding the general purpose of this project - detecting objects in the public landscape. Waste detection is the main issue found in most cities, but the amount of publicly available data related to this task is very small, particularly annotated data with an urban background. As discussed in the previous chapter, the images' context and the dataset's class balance have a big impact on the performance of computer vision models, so it is crucial to provide a significant amount of data with as much information as possible.

On that note, this work provides a compilation of four state-of-the-art datasets that fit the requirements of object detection in an urban environment. In addition, a completely new dataset of 3000 annotated images from the city of Funchal (in Madeira) was created, and it can be used as a foundation for future work or similar use cases in a different city. Nonetheless, it is not guaranteed that it will have the same performance in a different testing set without further training.

The best models trained during this work were able to achieve accuracy values greater than the corresponding state-of-the-art, with an overall mAP@0.5 above 0.78 for all classes included in the corresponding datasets. Even though these models can be further improved, they are already capable of accurately annotating the majority of the classes in pictures from Funchal. This significantly reduced the time required to annotate an image and incorporate it into the new training dataset, enabling a streamlined annotation process for future iterations.

7.2 Final Remarks

Overall, six different models were used, and more than twenty training iterations were performed during the development of this work. The training process was divided into two different tasks: detecting people and license plates, with the goal of ensuring privacy by design; and detecting waste containers, waste plastic bags, cardboard, traffic signs, graffiti, and billboards. Considering the achieved results, the proposed model for each task is the YOLOv7-tiny, and the YOLOv7-e6e, respectively.

In the first task, the highest overall mAP@0.5 achieved by the YOLOv7-tiny was 0.813, with a dataset of around 2200 images. On the other hand, the highest overall mAP@0.5 achieved in the second task was 0.802 by the EfficientDet-D4 model, with a dataset of more than 5000 images. However, after testing on pictures from Madeira, the model with the best performance for this task was in fact the YOLOv7-e6e, despite its slightly lower mAP@0.5 of 0.781.

The most relevant classes that achieved an accuracy lower than the average were the cardboard, plastic bag, and graffiti classes, mainly because of their heterogeneity. Regardless, when testing the models on images from Madeira, these classes were mostly well-detected, highlighting the possible success of the project in a real-world scenario.

This work has shown that good results can be achieved without a big investment or massive amounts of data. Also, by only relying on open-source models that can be accessed by anyone, it has shown that it is possible to tackle such a specific problem without needing to design and build a new customized model, especially when a model such as YOLO is so easy to use and increasingly better documented and maintained.

On the other hand, the initial stages of this project encountered a significant setback due to the time spent on training models with incorrect annotations. This could have been better managed by thoroughly analyzing every dataset from the beginning. Additionally, the incompatibility between the YOLOv7 models and the corresponding conversion to edge computing should have been avoided or at least detected earlier with better planning.

The lack of updated documentation and the challenges encountered when using EfficientDet, compared to the more user-friendly YOLO, extended the learning process. Furthermore, the larger training time required for an EfficientDet model - more than 20 hours, considering the EfficientDet-D4 - also limited the number of training iterations and subsequently impacted the quality of the obtained results. Nonetheless, despite not achieving the same performance as YOLOv7-e6e, more complex and accurate EfficientDet models with a larger input size should be superior to YOLOv7 models for this use case, assuming a class-balanced dataset.

The implementation of the k-folds cross-validation technique, despite its increased training time for the same number of epochs, consistently yielded more reliable results. This can be attributed to the class balancing function that was used in this training process, which enhanced the performance of the models and its correlation with the dataset's quality. Adopting this technique as the primary approach would have been more appropriate to surpass the limitations of the hold-out technique.

The detection of the billboard class was slightly overlooked in comparison to other classes due to this work's time constraints and the limited data quality available for this class. Expanding the selection of images with similar annotations from the "Open Images" dataset and implementing data augmentation could have improved the performance of this class. The cardboard and plastic bag classes could have also benefited from data augmentation to improve the detection of the different stances in which these objects can be found on the streets. However, this would have resulted in longer data pre-processing.

Even though some challenges and opportunities for improvement were encountered, this work successfully achieved the majority of its goals and demonstrated the possibilities of object detection in the public landscape using computer vision techniques. Future endeavors in this field can build upon these findings to further enhance the accuracy of object detection systems for real-world applications.

7.3 Future Work

Future work includes experimenting with the latest version of YOLO - YOLOv8 - to compare results with the previous models and to correctly convert the models in order to perform inference tests on an Android device. In contrast to what was discussed about YOLOv7 in section 6.3.

Another possible improvement lies in diversifying the datasets used for training the license plate class, comprising both newer and older standards that can still be observed on the streets of Portugal. This would enhance the model's ability to accurately detect and recognize different variations. Additionally, datasets with damaged traffic signs and billboards could also be collected to enable the detection of damages in both these classes.

There are still more than 13000 unannotated images left in Madeira's dataset. Hence, the models trained during this work can be used as an annotating tool in future iterations to improve the training dataset available.

Moreover, to better address overfitting issues and improve dataset class balance management, the k-folds cross-validation technique may be used from the beginning instead of hold-out, since it produces more realistic and reliable results that don't rely on random splits of data. The hold-out technique with data balancing could also be explored as a more time-efficient alternative to achieve similar benefits in terms of class balance.

Lastly, exploring the integration of object segmentation techniques alongside object detection could yield even more precise results. By accurately delineating the boundaries of objects, this approach would help avoid the issue of overlapping bounding boxes, which can frequently occur when dealing with piled waste bags and cardboard.

Addressing these topics can further improve this object detection system, leading to enhanced accuracy, adaptability, and robustness in real-world scenarios.

Appendix A

Evaluation Metrics

Complementary data for a better understanding of section [2.1](#).

A.1 Evaluation Metrics for Binary Classification

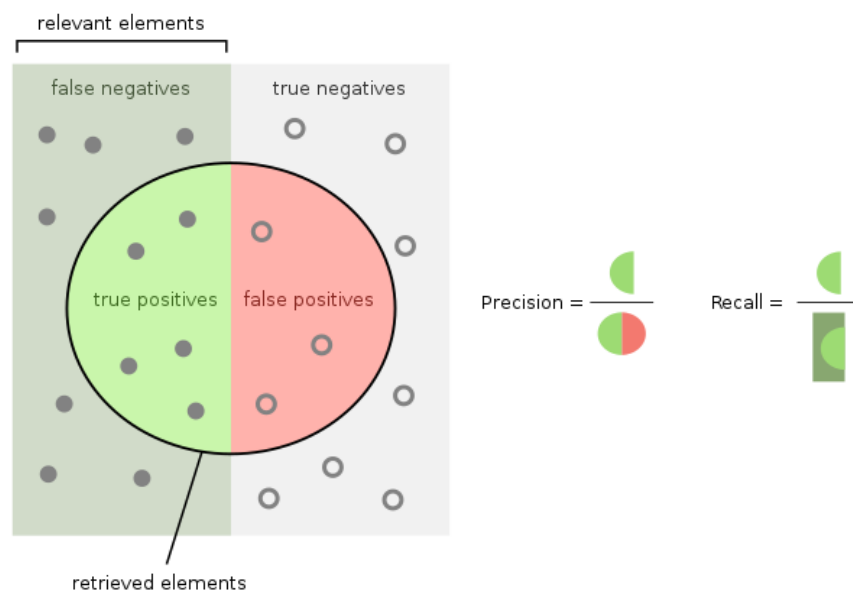


Figure A.1: Visual representation of the precision and recall metrics for binary object classification (Adapted from [\[7\]](#))

A.2 Evaluation Metrics for Non-Binary Classification

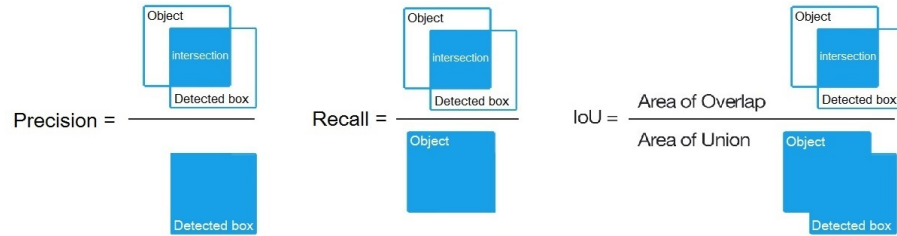


Figure A.2: Visual representation of the precision, recall and IoU metrics for non-binary object classification [8]

A.3 Confusion Matrix

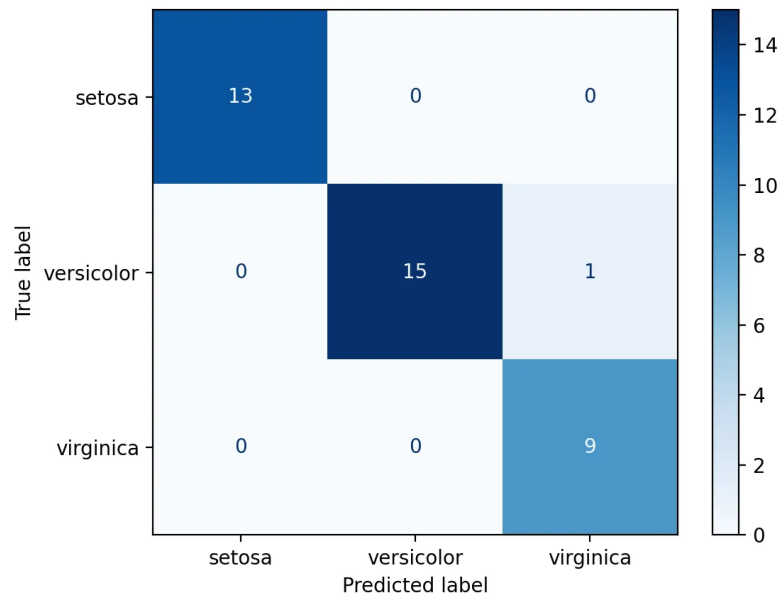


Figure A.3: An example of a confusion matrix with three different classes of flowers [9]

Appendix B

Object Detection Performance for Different Applications

Complementary data for a better understanding of chapter 3.

B.1 Datasets' Representative Images

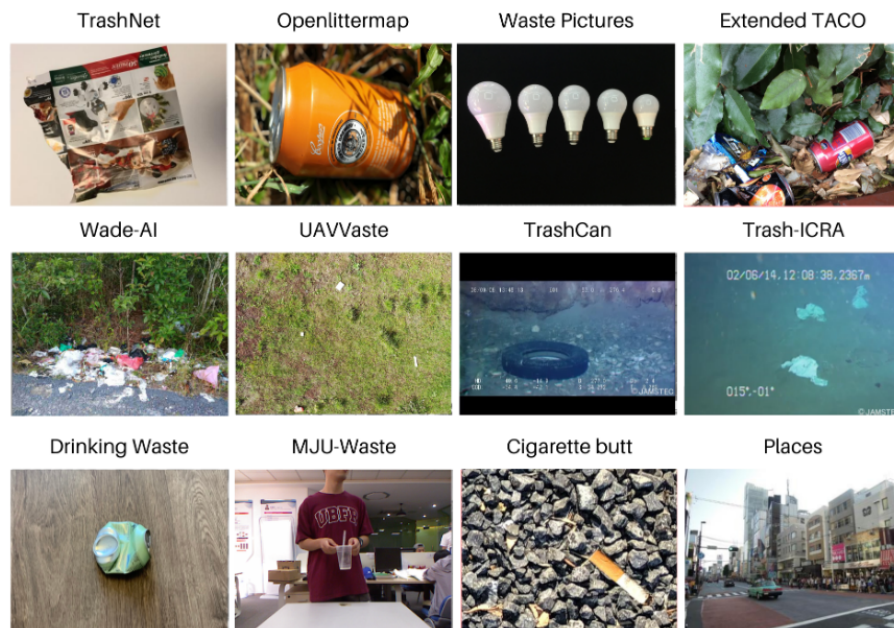


Figure B.1: Representative images from each dataset [10]

B.2 Pseudo-labelling Technique

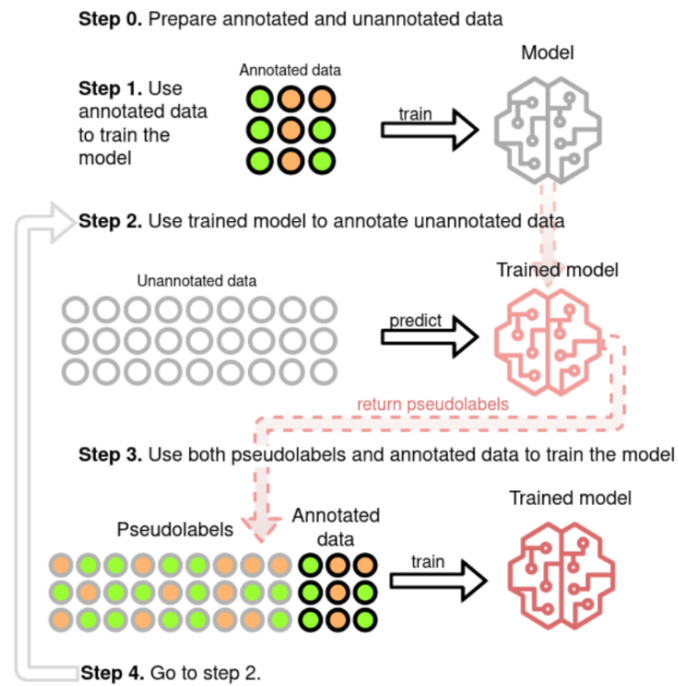


Figure B.2: Flowchart for pseudo-labeling semi-supervised learning technique [10]

B.3 Image Distribution per Dataset in Detect-waste+

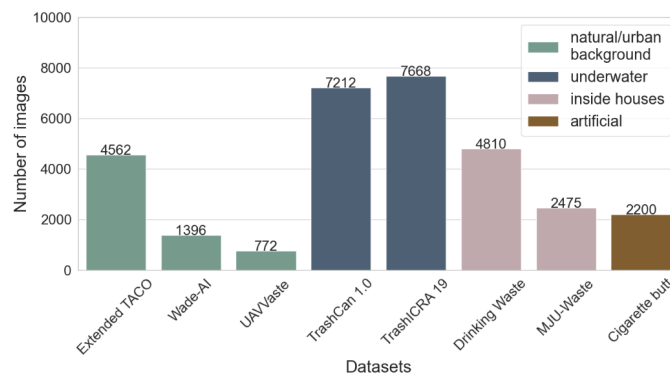


Figure B.3: Image distribution per class in classify-waste+ [10]

B.4 Image Distribution per Class in Extended TACO

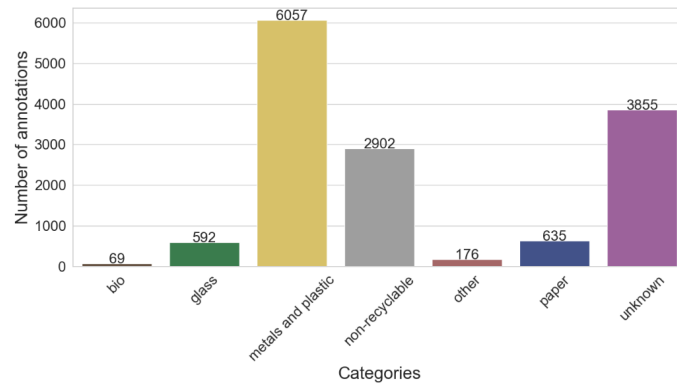


Figure B.4: Image distribution per class in Extended TACO [10]

B.5 Image Distribution per Class in classify-waste+

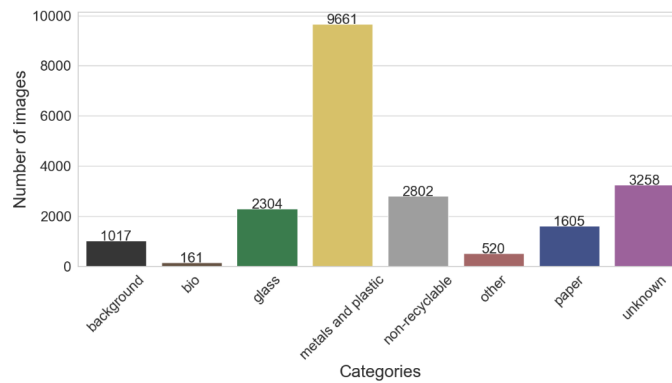


Figure B.5: Image distribution per class in classify-waste+ [10]

B.6 Confusion Matrix for the EfficientNet-B2 Model

Confusion Matrix

Predicted label \ True label	background	bio	glass	metals and plastic	non-recyclable	other	paper	unknown litter
background	296	1	1	3	0	2	1	2
bio	3	20	1	4	6	2	0	3
glass	0	2	381	35	26	1	2	16
metals and plastic	3	2	42	1374	186	11	73	260
non-recyclable	0	2	7	66	369	4	36	84
other	0	2	1	8	11	96	4	8
paper	1	1	1	22	26	3	246	23
unknown litter	1	2	24	68	91	16	34	431

Figure B.6: Confusion matrix for EfficientNet-B2 with weighted sampler and pseudo-labelling per batch [10]

B.7 Bounding Box Sizes for TACO-10

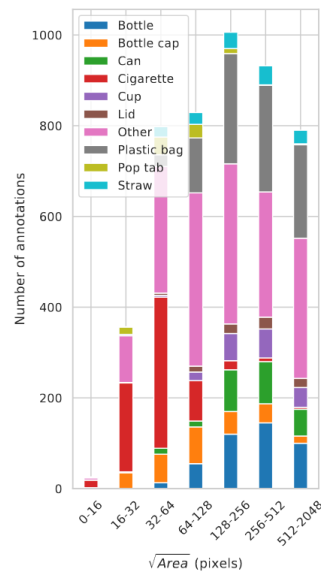


Figure B.7: Histogram of bounding box sizes per category for TACO-10 [11]

B.8 Confusion Matrices for Mask-RCNN

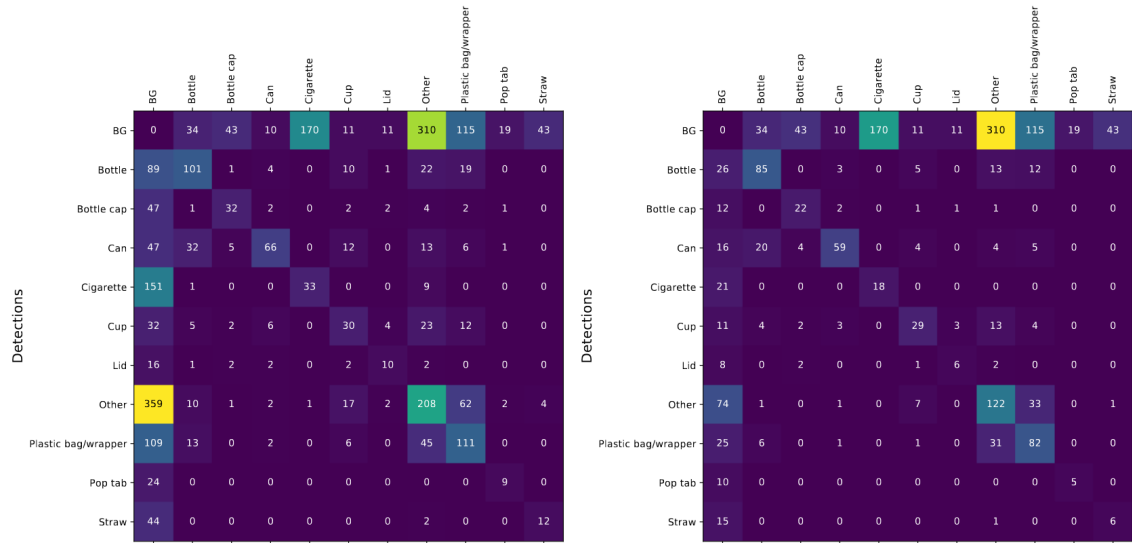


Figure B.8: Confusion matrices for matches with $IoU > 0.5$. (Left) detections with $ratio_score > 10$, (Right) detections with $ratio_score > 50$. BG represents background. [11]

Appendix C

Complementary Training Tools

Complementary data for a better understanding of chapter 5.

C.1 K-Folds Cross-Validation Python Script

Algorithm 1 Script created to train YOLOv7 models with k-folds cross-validation

```
1:  $k \leftarrow 5$ 
2:
3: Split all images and corresponding annotation files into k folds, using the StratifiedKFold()
   function
4:
5: for  $n := 1$  to  $k$  step 1 do
6:
7:     Merge every fold except fold n into the training set. Leave fold n as the validation set of
       this iteration's dataset
8:
9:     Create two .txt files - one with the path for every image in the training set and another for
       the validation set
10:
11:     Create a .yaml file with the path for both .txt files, the number of classes to be detected,
        and their names
12:
13:     Run the train.py script with the corresponding .yaml file to train the model for this fold
14:
15:     Run the test.py script with the corresponding .yaml file to validate the model for this fold
16:
17: end for
18:
19: Calculate the average accuracy values achieved in step 15 for each fold
```

References

- [1] Ashish Jaiswal, Ashwin ramesh babu, Mohammad Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9:2, 12 2020. doi:10.3390/technologies9010002.
- [2] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. 7:128837–128868. Conference Name: IEEE Access. doi:10.1109/ACCESS.2019.2939201.
- [3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. URL: <http://arxiv.org/abs/1512.03385>.
- [4] Mingxing Tan, Ruoming Pang, and Quoc V. Le. EfficientDet: Scalable and efficient object detection. URL: <http://arxiv.org/abs/1911.09070>.
- [5] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. Object detection with deep learning: A review. 30(11):3212–3232. Conference Name: IEEE Transactions on Neural Networks and Learning Systems. doi:10.1109/TNNLS.2018.2876865.
- [6] Maarten Sukel, Stevan Rudinac, and Marcel Worring. Urban object detection kit: A system for collection and analysis of street-level imagery. In *Proceedings of the 2020 International Conference on Multimedia Retrieval*, pages 509–516, 2020.
- [7] F-score. Page Version ID: 841695054. URL: <https://en.wikipedia.org/w/index.php?title=F-score&oldid=841695054>.
- [8] Alexey. Yolo v4, v3 and v2 for windows and linux. Accessed: 2023-01-01. URL: <https://github.com/AlexeyAB/darknet>.
- [9] Serafeim Loukas PhD. How to estimate FP, FN, TP, TN, TPR, TNR, FPR, FNR & accuracy for multi-class data in python in 5... Accessed: 2023-06-19. URL: <https://shorturl.at/fJSU5>.
- [10] Sylwia Majchrowska, Agnieszka Mikołajczyk, Maria Ferlin, Zuzanna Klawikowska, Marta A. Plantykw, Arkadiusz Kwasigroch, and Karol Majek. Waste detection in pomera-nia: non-profit project for detecting waste in environment. URL: <http://arxiv.org/abs/2105.06808>.
- [11] Pedro F. Proença and Pedro Simões. TACO: Trash annotations in context for litter detection. URL: <http://arxiv.org/abs/2003.06975>.
- [12] Asif Iqbal Middy, Debjani Chattopadhyay, and Sarbani Roy. Garbage detection and classification using faster-RCNN with inception-v2. In *2021 IEEE 18th India Council International Conference (INDICON)*, pages 1–6. ISSN: 2325-9418. doi:10.1109/INDICON52576.2021.9691547.

- [13] What is deep learning? | IBM. Accessed: 2023-06-19. URL: <https://www.ibm.com/topics/deep-learning>.
- [14] Vladimir Lyashenko. Cross-validation in machine learning: How to do it right. Accessed: 2023-01-03. URL: <https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right>.
- [15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. URL: <http://arxiv.org/abs/1409.0575>.
- [16] The PASCAL visual object classes homepage. Accessed: 2022-12-27. URL: <http://host.robots.ox.ac.uk/pascal/VOC/>.
- [17] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common objects in context. URL: <http://arxiv.org/abs/1405.0312>.
- [18] COCO - common objects in context. Accessed: 2022-12-23. URL: <https://cocodataset.org/#download>.
- [19] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. URL: <http://arxiv.org/abs/1905.11946>.
- [20] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-CNN. version: 3. URL: <http://arxiv.org/abs/1703.06870>.
- [21] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. 88(2):303–338. URL: <http://link.springer.com/10.1007/s11263-009-0275-4>, doi: 10.1007/s11263-009-0275-4.
- [22] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. URL: <http://arxiv.org/abs/1612.08242>.
- [23] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. URL: <http://arxiv.org/abs/1804.02767>.
- [24] 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. URL: <http://arxiv.org/abs/2207.02696>.
- [25] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023. URL: <https://github.com/ultralytics/ultralytics>.
- [26] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788. IEEE. URL: <http://ieeexplore.ieee.org/document/7780460/>, doi:10.1109/CVPR.2016.91.
- [27] YOLO: Real-time object detection. Accessed: 2023-01-02. URL: <https://pjreddie.com/darknet/yolo/>.

- [28] Glenn Jocher. YOLOv5 by ultralytics. Accessed: 2023-01-02. URL: <https://github.com/ultralytics/yolov5>, doi:10.5281/zenodo.3908559.
- [29] Marek Kraft, Mateusz Piechocki, Bartosz Ptak, and Krzysztof Walas. Autonomous, on-board vision-based trash and litter detection in low altitude aerial images collected by an unmanned aerial vehicle. 13(5):965. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute. URL: <https://www.mdpi.com/2072-4292/13/5/965>, doi:10.3390/rs13050965.
- [30] Jungseok Hong, Michael Fulton, and Junaed Sattar. TrashCan: A semantically-segmented dataset towards visual detection of marine debris. URL: <http://arxiv.org/abs/2007.08097>.
- [31] Michael Fulton, Jungseok Hong, Md Jahidul Islam, and Junaed Sattar. Robotic detection of marine litter using deep visual detection models. URL: <http://arxiv.org/abs/1804.01079>, doi:10.48550/arXiv.1804.01079.
- [32] Tao Wang, Yuanzheng Cai, Lingyu Liang, and Dongyi Ye. A multi-level approach to waste object segmentation. 20(14):3816. Number: 14 Publisher: Multidisciplinary Digital Publishing Institute. URL: <https://www.mdpi.com/1424-8220/20/14/3816>, doi:10.3390/s20143816.
- [33] Drinking waste classification. Accessed: 2022-12-30. URL: <https://www.kaggle.com/datasets/arkadiyhacks/drinking-waste-classification>.
- [34] Notice: WADE has been superseded by TrashAI which can be found here: <https://github.com/code4sac/trash-ai>. Accessed: 2022-12-22. URL: <https://github.com/letsdoitworld/wade-ai>.
- [35] Cigarette butt dataset. Accessed: 2022-12-30. URL: <https://www.immersivelimit.com/datasets/cigarette-butts>.
- [36] Mindy Yang and Gary Thung. TrashNet - classification of trash for recyclability status. URL: <https://cs229.stanford.edu/proj2016/report/ThungYang-ClassificationOfTrashForRecyclabilityStatus-report.pdf>.
- [37] Seán Lynch. OpenLitterMap.com – open data on plastic pollution with blockchain rewards (littercoin). 3(1):6. URL: <https://opengeospatialdata.springeropen.com/articles/10.1186/s40965-018-0050-y>, doi:10.1186/s40965-018-0050-y.
- [38] waste_pictures. Accessed: 2022-12-30. URL: <https://www.kaggle.com/datasets/wangziang/waste-pictures>.
- [39] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. 40(6):1452–1464. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence. doi:10.1109/TPAMI.2017.2723009.
- [40] ODK / ODK stack · GitLab. Accessed: 2023-01-02. URL: <https://gitlab.com/odk/odk-stack>.

- [41] Hardik Vasa. Google images download. Accessed: 2022-12-29. URL: <https://github.com/hardikvasa/google-images-download>.
- [42] Dong-Hyun Lee. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. URL: <https://shorturl.at/DFLY0>.
- [43] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. URL: <http://arxiv.org/abs/2010.04159>, doi:10.48550/arXiv.2010.04159.
- [44] Mask r-CNN for object detection and segmentation. Accessed: 2022-12-31. URL: https://github.com/matterport/Mask_RCNN.
- [45] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-CNN. URL: <http://arxiv.org/abs/1903.00241>.
- [46] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826. IEEE. URL: <http://ieeexplore.ieee.org/document/7780677/>, doi:10.1109/CVPR.2016.308.
- [47] Adel Ammar, Anis Koubaa, Mohammed Ahmed, Abdulrahman Saad, and Bilel Benjdira. Aerial images processing for car detection using convolutional neural networks: Comparison between faster r-CNN and YoloV3. 10(7):820. URL: <http://arxiv.org/abs/1910.07234>, doi:10.3390/electronics10070820.
- [48] K. R. Akshatha, A. Kotegar Karunakar, Satish B. Shenoy, Abhilash K. Pai, Nikhil Hunjanal Nagaraj, and Sambhav Singh Rohatgi. Human detection in aerial thermal images using faster r-CNN and SSD algorithms. 11(7):1151. Number: 7 Publisher: Multidisciplinary Digital Publishing Institute. URL: <https://www.mdpi.com/2079-9292/11/7/1151>, doi:10.3390/electronics11071151.
- [49] Open source data labeling. Accessed: 2023-06-02. URL: <https://labelstud.io/>.
- [50] Open Images V7. Accessed: 2023-05-24. URL: <https://storage.googleapis.com/openimages/web/index.html>.
- [51] voxel51/fiftyone. Accessed: 2023-06-01. URL: <https://github.com/voxel51/fiftyone>.
- [52] FiftyOne open source. Accessed: 2023-06-01. URL: <https://voxel51.com/fiftyone/>.
- [53] Roboflow: Give your software the power to see objects in images and video. Accessed: 2023-06-01. URL: <https://roboflow.com/>.
- [54] Kin-Yiu Wong. Official YOLOv7, May 2023. Accessed: 2022-07-06. URL: <https://github.com/WongKinYiu/yolov7>.
- [55] sklearn.model_selection.StratifiedKFold. Accessed: 2023-06-07. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html.

- [56] Dewan Fayzur Rahman. Garbage object detection (alpha). Accessed: 2023-02-23. URL: <https://github.com/dfayzur/garbage-object-detection>.
- [57] Patrikakis Charalampos, Kasnesis Panagiotis, Toumanidis Lazaros, and Tzitamidis Alexandros. STORM graffiti/tagging detection dataset. URL: <https://zenodo.org/record/3238357>, doi:10.5281/zenodo.3238357.
- [58] Ross Wightman. EfficientDet (PyTorch). Accessed: 2023-05-26. URL: <https://github.com/rwightman/efficientdet-pytorch>.
- [59] Ultralytics. Android. Accessed: 2023-06-10. URL: <https://docs.ultralytics.com/hub/app/android>.