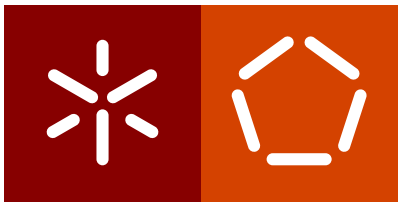


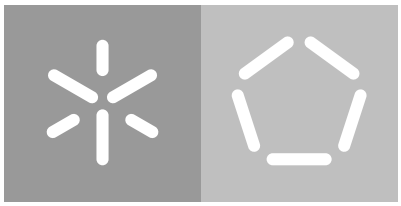
Universidade do Minho
Escola de Engenharia
Departamento de Informática

Nuno Barbosa Leão Beça e Silva

Prediction System

for Municipal Waste Containers

January 2024



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Nuno Barbosa Leão Beça e Silva

Prediction System

for Municipal Waste Containers

Master dissertation
Master Degree in Informatics Engineering

Dissertation supervised by
Pedro Rangel Henriques
Nuno Feixa Rodrigues

January 2024

AUTHOR COPYRIGHTS AND TERMS OF USAGE BY THIRD PARTIES

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

Nuno Silva

ACKNOWLEDGEMENTS

Throughout this thesis, there have been invaluable contributions that have significantly facilitated the research process. Firstly, I express my sincere gratitude to my thesis supervisors, Professor Pedro Rangel and Professor Nuno Rodrigues. Our regular meetings provided essential structure and guidance, and their determined support, receptivity, and accessibility for addressing doubt and defining research directions greatly accelerate the completion of this Master's project.

I also extend my appreciation to friends who played a pivotal role in the initial collection of container images. Their assistance was instrumental in creating the foundational versions of the dataset, which proved essential for training the container detection model.

I would like to acknowledge the significant contribution of André Torneiro, an expert in the Tidy City project. André's availability and invaluable insights, particularly in the domain of object detection, provided me with essential guidance during the model training process.

Lastly, I extend my gratitude, though indirectly, to my parents, sister, girlfriend, and friends. Their extraordinary support, motivation, and encouragement have been instrumental in boosting my resilience and motivation throughout this journey. Their contribution, though less perceptible, was no less vital.

ABSTRACT

Recycling stands as one of the most effective contemporary practices for pollution prevention. Through the process of recycling, a reduction in our reliance on finite natural resources is achieved, concurrently leading to energy conservation, decreased carbon dioxide emissions, and economic savings. In the context of the European Union, it is noteworthy that Portugal currently registers one of the lowest recycling rates. Consequently, it becomes imperative for the nation to commit towards accomplish the European objective of recycling all single-use packaging materials. A significant strategy to boost these recycling rates involves the widespread deployment of small, medium or large capacity waste containers, typically ranging from 120 liters to 360 liters, across municipalities. However, the efficient management of these containers necessitates a consistent and meticulous approach by waste collection entities. Presently, the methodology employed in this regard is antiquated, characterized by waste collection teams manually inspecting each container within their designated areas to check their fill status. This labor-intensive process poses inherent inefficiencies and challenges. The primary objective of this master's project involves the development of a system capable of detecting and classifying urban waste containers. This goal holds promising applications in the domain of waste management, potentially facilitating the generation of daily collection routes in the future. Images for this study were sourced from individual contributors, from Street View feature in Google Maps and a project known as Tidy City, which gathers various items, including containers, from a designated municipality. Subsequently, a model was constructed with the ability to discern and categorize a specific container based on the type of waste it accommodates, the configuration of the container (e.g., 4 wheels, 2 wheels), and the condition of its lid (open, closed, or full). Additionally, the model demonstrates proficiency in identifying and classifying waste materials in close proximity to the container.

Keywords: Waste Management, Object Detection, Instance Segmentation

RESUMO

A reciclagem é uma das práticas contemporâneas mais eficazes para a prevenção da poluição. Através do processo de reciclagem, consegue-se uma redução da nossa dependência de recursos naturais finitos, conduzindo simultaneamente à conservação de energia, à diminuição das emissões de dióxido de carbono e a poupanças económicas. No contexto da União Europeia, é de salientar que Portugal regista atualmente uma das mais baixas taxas de reciclagem. Consequentemente, torna-se imperativo que o país se empenhe em cumprir o objetivo europeu de reciclar todos os materiais de embalagem de utilização única. Uma estratégia significativa para aumentar estas taxas de reciclagem envolve a implantação generalizada de contentores de resíduos de pequena, média ou grande capacidade, normalmente entre 120 litros e 360 litros, em todos os municípios. No entanto, a gestão eficiente destes contentores exige uma abordagem consistente e meticulosa por parte das entidades de recolha de resíduos. Atualmente, a metodologia utilizada nesta matéria é antiquada, caracterizada por equipas de recolha de resíduos que inspecionam manualmente cada contentor dentro das suas áreas designadas para verificar o seu estado de enchimento. Este processo de trabalho intensivo apresenta ineficiências e desafios inerentes. O objetivo principal deste projeto de mestrado é o desenvolvimento de um sistema capaz de detetar e classificar contentores de lixo urbano. Este objetivo tem aplicações promissoras no domínio da gestão de resíduos, facilitando potencialmente a criação de rotas de recolha diárias no futuro. As imagens para este estudo foram obtidas de colaboradores individuais, através da funcionalidade de Street View do Google Maps e de um projeto conhecido como Tidy City, que recolhe vários objetos, incluindo contentores de resíduos sólido urbanos, de um município designado. Posteriormente, foi construído um modelo com a capacidade de distinguir e categorizar um contentor específico com base no tipo de resíduos que acomoda, na configuração do contentor (por exemplo, 4 rodas, 2 rodas), e no estado da sua tampa (aberta, fechada ou cheia). Para além disso, o modelo demonstra proficiência na identificação e classificação de materiais residuais nas proximidades do contentor.

Palavras-Chave: Gestão de Resíduos Sólido-Urbanos, Deteção de Objetos , Segmentação de Instância

CONTENTS

LIST OF FIGURES

ACRONYMS

A

ANFIS Adaptive Neuro Fuzzy Inference System.

ANN Artificial Neural Network.

B

BPNN Back Propagation Neural Network.

C

CNN Convolutional Neural Networks.

COCO Common Objects in Context.

CPU Central Processing Unit.

CUDA Compute Unified Device Architecture.

D

DT Decision Tree.

F

FLOPS FLoating-point Operations Per Second.

G

GBRT Gradient Boosted Regression Trees.

GPU Graphics Processing Unit.

GRNN General Regression Neural Network.

K

KNN K-Nearest Neighbors.

L

LSTM Long Short-Term Memory.

M

ML Machine Learning.

MSWM Municipal Solid Waste Management.

O

ONNX Open Neural Network Exchange.

R

RAM Random Access Memory.

RF Random Forest.

S

SVM Support Vector Machine.

SVR Support Vector Regression.

1

INTRODUCTION

Waste recycling is the most effective practice to prevent pollution, reduce dependence on natural resources, save energy and reduce CO2 emissions, while promoting savings of financial resources (Agency, 2022)(Koop, 2022).. Portugal has one of the lowest waste recycling rates of all EU countries, recycling only 21 percent (in 2022), when the European goal is 55 percent (Soares, 2022a) (Soares, 2022b). One of the most effective measures to increase the recycling rate is the provision of smaller capacity (120L , 240L and 360L) public waste containers to serve a small set of nearby households (C40 Cities Climate Leadership Group, 2019). Presently, waste collection teams operate with notable inefficiency. Their current methodology involves physically inspecting container collection sites to manually confirm the fill status of urban waste containers. This approach presents the risk of unjustified trips to containers that may not necessitate servicing due to being inadequately filled. The principal objective of this master's thesis is to detect and classify various types of urban waste containers. Furthermore, the research aim to determinate their status (whether they are full, open, or closed), and determine the composition of adjacent waste materials proximate to the containers.

1.1 OBJECTIVES

The principal objective of this Master's project is to develop a model with the ability to detect and classify municipal waste container by analyzing video frames or images provided by a specific municipality.

Concurrently, several other pertinent objectives are pursued within this plan:

- The identification of container type by waste (paper, plastic, among others) represents a key facet, with the model possessing the capability to differentiate among diverse categories of waste containers.
- The model should be capable of detecting the status of the container's lid, discerning between open, closed, or overfilled waste containers.
- The identification of adjacent waste types in proximity to the containers is an integral aspect, by giving the model's ability to determine the specific material present in the proximity of the waste containers.

These objectives are inherently interlaced with a shared contemporary imperative: the capital goal of elevating the recycling rate to its maximum potential.

1.2 RESEARCH HYPOTHESIS

This master's thesis aims to develop a automatic detect system capable of detecting urban wastecontainers, their state (open, closed or full) and waste residues in images.

1.3 METHODOLOGY

The progression of this master's thesis will be organized into three principal phases:

- **Literature Review and Solution Design Phase** - In the initial phase, a comprehensive review of pertinent literature related to waste container monitoring systems, predictive models for waste container fill levels, and the identification of waste containers will be undertaken. Simultaneously, a software architecture solution will be concluded.
- **Development Phase** - The second phase encompasses the development process. It commences with the construction of datasets essential for model training. Once the dataset reaches a sufficient level of robustness, the model training phase commences. Upon the successful completion of model training, the final phase begin.

- **Validation and Results Analysis Phase** - The last phase is dedicated to the validation and comprehensive analysis of the outcomes acquired during the training process. This phase serves to assess the efficacy and performance of the developed model.

It's imperative to emphasize that proceeding to the third phase does not signify the completion of the second phase. In certain instances, a return to the second phase may be essential to enhance the outcomes of the third phase, ensuring ongoing refinement and improvement of the results.

1.4 DOCUMENT STRUCTURE

This master's thesis comprises a total of seven chapters. Chapter one serves as an introduction, aligning the project's objectives and anticipated achievements. In the second chapter, a historical context is provided, offering background information on Municipal Solid Waste Management (MSWM) and other pertinent subjects relevant to the Master's project.

Following this, the third chapter offers an in-depth exploration of the state of the art in the field, giving insights into relevant companies, image annotation tools, object identification algorithms, and related areas of interest.

Chapter four presents a detailed system architecture, offering practical guidance for executing the project. Chapters five and six register the project's various stages, including decision-making processes and an analysis of the results collected throughout the project's duration.

Lastly, the document culminates in chapter seven, where the project concludes.

2

BACKGROUND AND CONTEXT

In the upcoming chapter, a thorough and comprehensive examination of the current conditions and theoretical support to Municipal Solid Waste Management (MSWM) will be presented. This undertaking will involve a detailed exploration of the contextual elements and theoretical foundations that characterize the contemporary landscape of MSWM.

2.1 HISTORY OF THE MUNICIPAL SOLID WASTE MANAGEMENT

As early as 2000 BC, solid waste management systems were primarily guided by religious beliefs, aesthetic considerations, and concerns for public health. By 500 BC, the ancient Greeks had established what could be considered the Western world's first officially recognized 'municipal dumpsites' and even established the first known decree prohibiting the disposal of garbage in streets. Inadequate waste management in urban areas contributed to the spread of diseases, including the Black Death. Various methods were employed for waste disposal, such as using waste as fuel for indoor burners or burning it in fireplaces and outdoor bonfires. Additionally, food waste was often repurposed as animal feed, particularly for swine. Some waste was disposed of in open bodies of water like ponds, bogs, lakes, rivers, and even the ocean.

The advent of the industrial revolution significantly increased the urban population in Europe and the United States. However, until the latter half of the nineteenth century, the United States fall off behind Europe in terms of population growth and the associated challenges. Technology, coupled with industrialization, played a pivotal role in shaping the early evolution of waste management. Industrialization attracted populations to urban centers, resulting in the generation of substantial amounts of waste. The sanitation issues ascending from this waste necessitated the establishment of organized municipal waste management systems. These challenging conditions ultimately led to the creation of structured municipal sanitation services.

Given the magnitude of these issues, scientists worldwide began to explore advanced technologies to address them. Machine learning (ML) algorithms, known for their ability to model complex nonlinear processes, have gained traction in recent years as a means to

enhance municipal solid waste management (MSWM) and contribute to the sustainable development of the environment.Today, the use of these algorithms in this field can be seen in composting control and optimization processes, heating value forecasts in incineration processes, leachate generation forecasts, among others. (TARASFOUNDATION, 2020)(de Souza Marotta Alfaia et al., 2017) (Louis, 2004).

2.2 MACHINE LEARNING ALGORITHMS IN MSWM

In order to choose the most suitable machine learning algorithm for MSWM, extensive research was done, and below is a description of this research, as well as an explanation of the algorithms mentioned.

2.2.1 *Artificial Neural Network*

An artificial neuron has been created to mimic the action of a biological neuron, i.e. to accept several different types of signals from other neighboring neurons. These signals are then processed and pre-defined. Depending on the result of this processing, the neuron decides whether to give the output signal or not. Normally, this signal can be 0 or 1, or a real value between these two numbers. These neurons are usually made up of two parts, the first of which is the weight function, which decides what proportion of the incoming signal should be transmitted to the body of the neuron. The second part is a transfer function that transfers the signals to other neurons. Artificial neural networks (ANNs) can be composed of different numbers of neurons, i.e. the number of neurons, are ranging from tens of thousands to only as little as less than ten (1-3) (Zupan, 1994).

2.2.2 *Support Vector Machine*

The Support Vector Machine (SVM) was developed in 1995 by Cortes and Vapnik for binary classification. This algorithm searches for the optimal separation hyperplane between two classes by maximizing the margin between the points closest to the classes. Data points on the "wrong" side of the discriminated margin are weighted down to reduce their influence. If no linear separator can be found, these data points are projected into a higher-dimensional space, where the data poiunts become linearly separable (Meyer, 2001).

2.2.3 *Support Vector Regression*

The SVR algorithm operates on the same principles as SVM,however there is small diferences. First of all, because output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities. In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. But besides this fact, there is also a more complicated reason, the algorithm is more complicated therefore to be taken in consideration. However, the main idea is always the same: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated.

2.2.4 *Decision Tree*

A decision tree (DT),is a classifier expressed as a recursive partition of the space of instances. The decision tree consists of nodes that form a rooted tree, meaning it is a directed tree with a node called "root" that has no incoming edges. All the other nodes contain only one incoming edge. A node that contains outgoing edges is called a test node. The remaining nodes are called leaves (also known as decision nodes). In a decision tree, each internal node divides the space of instances into two or more sub-spaces according to the input values of a certain function. In the most frequent cases, each test considers a single attribute, such that the instance space is partitioned according to the attribute's value. In the case of numeric attributes, the condition refers to a range. (Rokach and Maimon, 2005).

2.2.5 *K-Nearest Neighbors*

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier that uses proximity to classify or predict the grouping of a single data point. While it can be used for either regression or classification problems, it is most commonly used as a classification algorithm, assuming that similar points can be found near one another. (IBM).

2.2.6 *Random Forest*

Random forest, as the name implies, is made up of a large number of individual decision trees. In the random forest, each decision tree predicts one class. The class with the most votes is chosen by the algorithm to be the model's final prediction. (Yiu, 2019).

2.2.7 *Convolutional Neural Networks*

A convolutional neural network (CNN) is a specialized type of deep learning algorithm specifically designed for tasks involving image recognition and processing. This network architecture comprises multiple layers, including convolutional layers, pooling layers, and fully connected layers.

The distinctive feature of CNNs lies in their convolutional layers, where filters are applied to the input image to extract various features such as textures, shapes, edges, and more. These layers play a crucial role in identifying patterns within the images.

Following the convolutional layers, the output is passed through pooling layers, which serve to downsample the feature maps, reducing the dimensionality while retaining the most significant information. This downsampling helps in simplifying the subsequent computations.

Ultimately, the output from these layers is passed through one or more fully connected layers, which play a pivotal role in making predictions or classifying an image based on the features extracted by the preceding layers. This architecture has proven to be highly effective in tasks related to image analysis and recognition, (GeeksforGeeks, 2023)

2.3 OBJECT DETECTION

Object detection is a deep learning technique that can recognize a variety of objects in images or videos.

Object detection is merely the recognition of an object through a bounding box or a segmentation of a certain instance, with respect to image classification, it is possible to categorize it (classify) as being an object or not in terms of probability.

Using neural networks as an example, when an image or video is provided as input, the output will be probability values for each class that is represented in the image or video (Patel, 2020).

The image 1 illustrates an instance of detecting two animals, specifically a dog and a cat. The algorithm is presented with these two images and, after processing through its layers, produces an output that identifies the desired object in the image. In this case, the identification is indicated by a bounding box annotation.

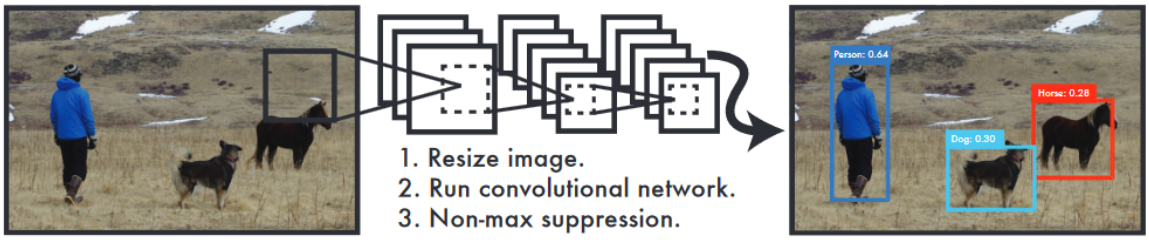


Figure 1: Object Detection with Bounding Box (Patel, 2020)

2.4 ANNOTATION TYPES

Nowadays, image annotation is done using three most common methods. These annotation techniques differ from each other, and are more or less suitable for different types of problems. The three most common image annotation techniques are image classification, object detection and object segmentation. These techniques will be explained in detail below.

2.4.1 *Image Classification*

Image classification is a method of image annotation that involves labeling or categorizing images based on any similarities they may have within a given category. This kind of work, meanwhile, can be challenging because the model's accuracy can be affected by background objects in the photos. It is crucial to expose the machine learning model to various pictures of

each class in order to better categorize in an effort to prevent these occurrences. In the actual world, this method is frequently used, particularly in the healthcare sector, to categorize organs, illnesses and injuries. Thus, these images aid medical personnel in the identification, management, and prevention of diseases (Sayedi, 2021).

Several categories can be used to classify images as seen in image 2,(MathWorks):

- **Binary Classification**- Where images from two classes are labeled, uses "clean" images to classify data because they clearly show an object in its frame without any background noise.
- **Multiclass Classification** - With the exception of the fact that there are more than two classes identifying the photos, multiclass classification is a type of image classification similar to the one described previously.
- **Multilabel Classification** - Unlike the previous two strategies, this one makes use of a deep learning model to forecast the likelihood that a given image belongs to a particular class. Each class in this model has a distinct binary classifier. For instance, this model might have the classes "Cat", "Not Cat", "Dog" and "Not Dog" for a dataset that aims to categorize an animal as either a dog or a cat.
- **Hierarchical Classification** - This method divides classification problems into two sub-problems. In the first step, a CNN algorithm is used to "propose" an informative region of the object. The region and its classification are retained and a parent class is generated based on the material or shape of the object. The second step consists of cropping and resizing the region obtained in the original image, "feeding" this new image into the CNN, (Sousa et al., 2019).

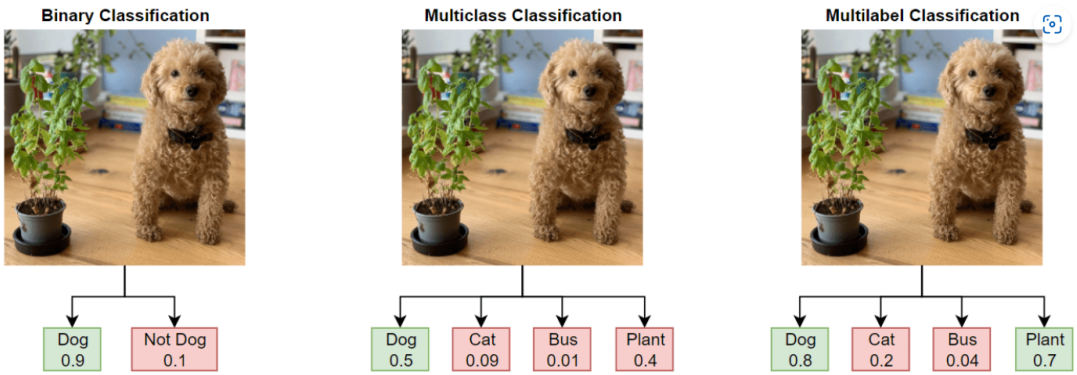


Figure 2: Types of image classification (MathWorks)

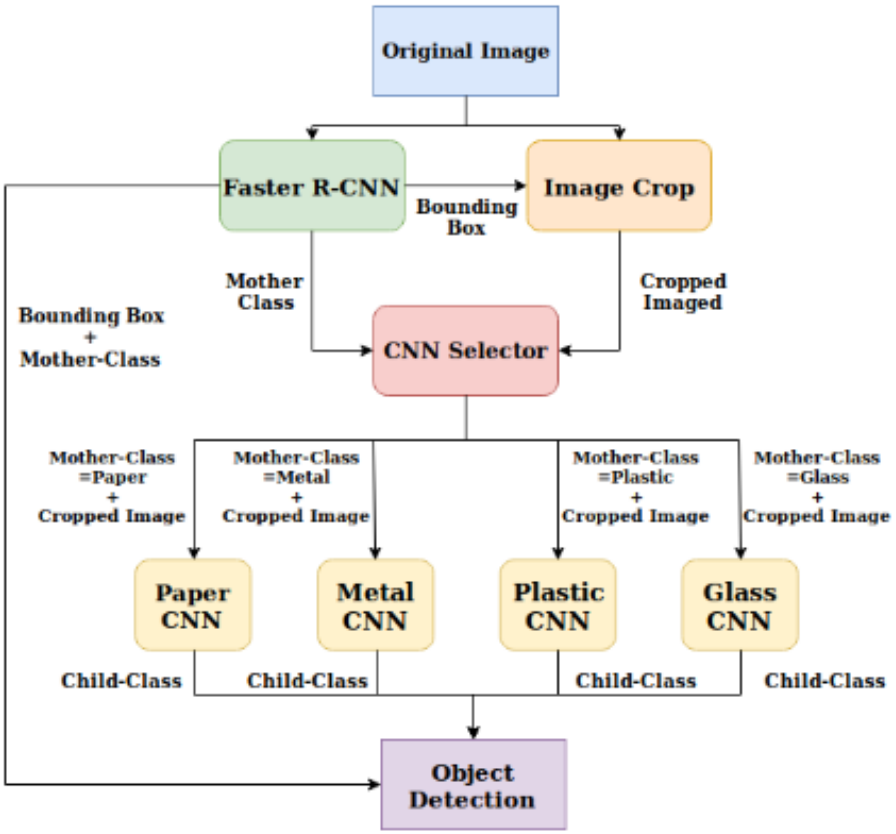


Figure 3: Hierarchical Classification(Sousa et al., 2019)

2.4.2 *Object Detection*

Object detection is a technique used to identify various objects within an image. This task is more complex compared to previous assignments because it involves dealing with multiple variables when identifying multiple objects in an image. Object detection differs from image classification in its ability to recognize several items within an image.

To accomplish object detection, bounding boxes are employed to delineate and identify these objects,bounding box are visible in the image 4, surrounding both the cat and the dog . These boxes are used to specify and instruct the model about the objects it should recognize, as they surround the objects of interest. Bounding boxes also enable the identification of an object based on its size, including its width and height.(Sayedi, 2021).

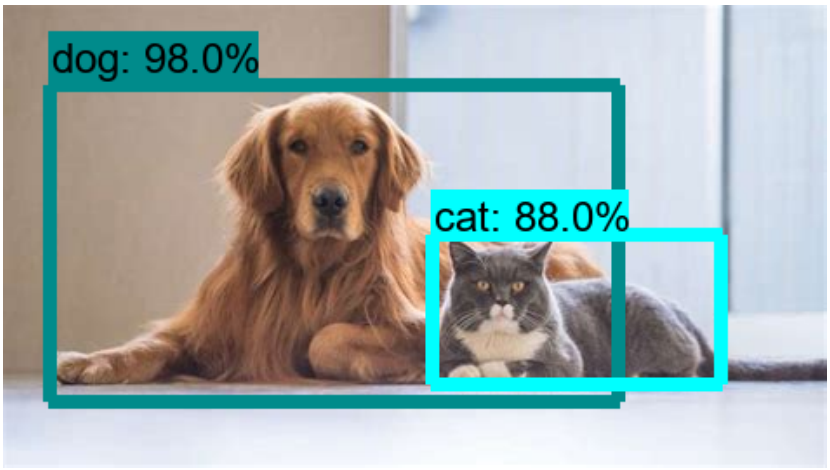


Figure 4: Example of object detection, the values correspond to the confidence of the model that the object corresponds to the described class.

2.4.3 *Object Segmentation*

Comparing this technique to the previous one, it represents a different approach. Instead of using bounding boxes to identify objects in images, this technique focuses on a pixel-level approach.

In this technique, each pixel of the object to be detected in the image is considered individually. These pixels function as boolean attributes, indicating whether they belong to the object or not. The outcome of this technique is a mask image, which is essentially a collection of pixels used to highlight specific attributes or regions within the image. (Sayedi, 2021).

Within this technique there are two major types of sub-techniques (image 5).

- **Semantic Segmentation** - This method involve giving the same color to the pixels that are recognized as being a part of the object while giving a different color to the pixels that do not.
- **Instance segmentation** - In contrast to the prior method, each set of pixels used to identify an object in this method is given a unique color.

Figure 5: Image Segmentation and its two types,(Sayedi, 2021).

3

STATE OF THE ART

In this chapter, an exploration of machine learning begins with an explanation of its fundamental concepts, methodologies, and implications in modern industries. Additionally, key machine learning algorithms will be introduced. The chapter will also provide insights into current industry use cases and relevant tools pertinent to the scope of this Master's project.

3.1 COMPARISON OF ML ALGORITHMS IN MSWM

The table 1 offers a comprehensive examination of the pros and cons associated with distinct algorithms within the scope of Municipal Solid Waste Management (MSWM). The table 1 has been extracted from a study authored by Xia et al. (2022).

As one examines the matter closely, it becomes evident that there are both advantages and disadvantages to be derived from employing various algorithms in the context of MSWM. Given the vast scope of MSWM, the selection of these algorithms should be dependent upon the project's established priorities. For instance, if the project places a priority on both high interpretability and efficiency, while disregarding the propensity for overfitting, the decision tree algorithm emerges as the preferred choice. To aid in the decision-making process for such projects, the provided table offers a means of evaluating and selecting suitable algorithms. However, it is imperative to acknowledge that this table serves as a condensed overview, and therefore, a more comprehensive analysis of the specific algorithm in question must be undertaken before reaching a final determination. **In the context of this master's project, significant attention should be given to analyzing the CNN algorithm due to its widespread utilization in object detection as well as its well-documented outcomes.**

	Algorithms	Advantages	Disadvantages
Traditional ML Algorithms	ANN	Suitable for any nonlinear relationship	Need a lot of parameters
		Strong robustness and fault tolerance	Lack of interpretability
	SVM/SVR	Suitable for small sample problems	Sensitive to missing data
		Avoid the local minima	Sensitive to kernel selection
		Low generalization error	
	DT	High interpretability	Prone to overfitting
		High efficiency	Ignore feature correlation
	KNN	No assumption for input data	Need large amount of calculation
		Not sensitive to outliers	Low accuracy
	ANFIS	Combine the advantages of neural network and fuzzy reasoning	Not suitable for higher dimensional features
	K-means	Easy to implement and fast convergence	Sensitive to noise
		Few parameters	Depend on the initialization of the cluster center
	RF	Reduce model variance	Not applicable to attribute data with different values
		No feature selection required	
	GBRT/GBDT	Rank feature importance	Difficult to train in parallel
DL algorithms	CNN	Able to extract features automatically	Need parameter tuning
			Need a lot of sample data
	RNN/LSTM	Effective for sequence data	Need large amount of calculation

Table 1: Advantages and disadvantages of different ML algorithms used in MSWM (Xia et al., 2022)

3.2 Comparison with the effectiveness of ML Algorithms in real situations

The table 2 refers to a study conducted by the author Xia et al. (2022) , where various algorithms are showcased in the context of Municipal Solid Waste Management (MSWM), demonstrating their respective use cases.

The presented table exhibits outputs derived from diverse inputs, accompanied by an estimated forecast and the corresponding algorithms employed for such prevision. Additionally, the table presents the pros and cons associated with these outputs. **Predominantly emerging in the table is the ANN algorithm, which is revealing its remarkable efficacy across a multitude of MSWM domains.**

Input feature	Output	Forecast period	Algorithm	Advantages	Disadvantages
Urban form, demographic and socioeconomic, seasonality related	Weekly municipal waste generation in New York, USA	Short-Term	GBRT, ANN	Analyze feature importance	Accuracy is not very high
Historical weekly waste generation	Weekly household waste generation rate	Short-Term	LSTM, ANN	Use multi-site models to increase amounts of historical data	Only choose ANN as ML comparison model
Education, occupation, income, house type	Weekly plastic waste generation rate in Dhanbad, India	Short-Term	ANN, SVM, RF	Predict a specific component of MSW	Low accuracy
Population, formally employed, unemployed and number of family units	30years MSW generation in Johannesburg , South Africa	Long-term	ANN, SVM	High accuracy	Easy to overfitting
Fraction of population over 45years, median personal income, employment rate, fraction of owned dwellings	Annual MSW generation in Ontario , Canada	Mid-Term	ANN, DT	Use feature selection techniques	Low accuracy
Population, solid waste collection frequency, maximum seasonal temperature and altitude	Seasonal MSW generation rate in Fars province, Iran	Short-Term	ANN	Use regularization to prevent overfitting	Lack of comparison with other ML algorithms
Waste generation in past twelve month	Monthly MSW generation in Logan, Australia	Short-term	SVM, ANN, ANFIS, KNN	High accuracy	Lack of comparison with other papers
Socio-economic parameters such as GDP population, condition of public infrastructure construction	Annual MSW generation in three regions of China	Mid-Term	ANN	Evaluate the effect of each predictor	Lack of comparison with other ML algorithms
GDP, rain, maximum temperature, population, household size, educated man, educated women, income, and the unemployment rate	Monthly and Seasonal municipal waste generation	Shot-Term	ANN, SVM, ANFIS	Make feature correlation analysis	The accuracy is not high
GDP per Capita, domestic material consumption, resource productivity	Annual MSW generation in 26 European countries	Mid-Term	BPNN, GRNN	Provide prediction for countries at different economic levels.	Lack of comparison with other ML algorithms

Table 2: Summary of ML algorithms used to predict MSW generation.

3.3 WASTE CONTAINER FILLING AND OTHER MSWM SOLUTIONS

In this section, an examination of several commercially available solutions enabling the restriction of access and the anticipation of when a municipal waste container reaches full capacity will take place. Additional strategies for increasing the recycling of waste materials will also be addressed.

3.3.1 *Detecting and classifying bus stop trash cans using camera-equipped public transit vehicles*

This study uses object detection algorithms in images to understand the state of filling of containers at bus stops. To do this, public transport vehicles are equipped with cameras that capture the state of these containers in real time. The algorithm used is Faster R-CNN with the ResNet+FPN backbone via detectron 2. It can be seen in more detail at, Storm and Mertz.

3.3.2 *SENSONEO*

The first example to be considered is SENSONEO, a company founded in 2017(sensoneo). SENSONEO specializes in real-time waste monitoring solutions and utilizes sensors embedded in waste containers to track their filling levels. This company offers a range of solutions that are all related to waste management in urban areas. An example of a sensor

for measuring the fill level of a municipal waste container can be seen in the image 6 (it's the object attached above it).

Figure 6: Container using Sensoneo technology

3.3.3 *Candam*

Other businesses are looking for ways to increase recycling rates in ways other than those listed above.

Candam, for example, employs containers that incorporate RecySmart technology (image 7). After the garbage has been deposited in their containers, this technology uses a refund system. It also employs an access control system to deposit garbage. In this way, they increase the rate of recycling and dumping garbage where it belongs(candam).

Figure 7: RecySmart technology by Candam

3.3.4 *Tidy City*

Tidy city is a project that aims to provide vehicles with mobile devices that can detect and categorize flaws in publicly accessible infrastructure. To reduce pollution and CO_2 emissions, photos will be collected everyday in real time and submitted to a server,the photos will then be processed and categorised using AI models enabled by the Urban Object Detection Kit. Furthermore, the system will be able to detect and record the GPS position of damage to outdoor advertising, lighting, and power supply infrastructure, as well as improper trash disposal scenarios (e.g., waste outside containers). It is vital to note that this initiative is still in the works and is now being tested in the Madeira archipelago (arditi).

3.4 Software used for image annotation

This section will present the results of the investigation of software with its features, ready to annotate, label or classify images.

3.4.1 *Roboflow*

Roboflow is a web platform that uses data gathering, prediction, preprocessing, and model training methods to enable users to create computer vision models more rapidly and

precisely. Users of this site can upload datasets, annotate, change the size and orientation of their pictures, adjust contrast, and do data augmentation. On this platform, models may also be trained.

It is crucial to note that although model training on this platform was evaluated for this Master's project, picture annotation was its primary application.

3.4.2 *Label Studio*

LabelStudio, is an open source data labeling tool that offers support for multiple projects, users, and data formats. Its utility lies in the ability to utilize predictions from a single machine learning model for data annotation. Moreover, once these predictions are validated, they can be exported in various formats to be seamlessly incorporated into machine learning algorithms. The Tidy City project used the potential of this software to identify a multitude of objects within the photos collected for their dataset. (Tkachenko et al., 2020-2022).

3.4.3 *CVAT.ai*

CVAT (Computer Vision Annotation Tool) is a tool for annotating both still and moving pictures. Users of this program can input their datasets and perform different annotating activities. Bounding box and other image segmentation methods can be used for annotation. Additionally, it provides the ability to export picture annotations in a number of formats, including json, Yolo, etc, (CVAT.ai Corporation, 2022)

3.5 Algorithms widely used in this domain

This section will present some commonly used algorithms. These algorithms are open-source, have adequate documentation, and can be easily implemented in somebody's code.

3.5.1 *YOLO*

YOLO ("You Only Look Once") (Karimi, 2021) is an ML algorithm that detects and recognizes objects in a video or image (in real time). This algorithm treats the task of identifying objects in images or videos as a regression issue and outputs probability that the discovered objects fall into a specific class. The algorithm uses CNN (Saha, 2018) to detect objects in real time. The method is true to its name, "just look once," since it only has to run once through a neural network in order to recognize objects, — in other words, it can anticipate the complete video or image in only one iteration.

YOLO has important advantages that can be applied to this Master's project:

- **Speed** - The feature of "only looking once" at the image or video speeds up detection because it can forecast things in real time.
- **High accuracy** - YOLO is a predictive technique that provides accurate results with minimal background errors.
- **Learning Capabilities**-The algorithm has excellent learning capabilities that enable it to learn the representations of objects and apply them in object detection.

Nowadays this algorithm is used in real situations, such as

- **Autonomous driving**- Here object detection is of greater importance to avoid collision with other cars, since no human driver is controlling the car.
- **Wildlife**-This algorithm is also used to detect various types of animals in forests, jungles, etc.
- **Security**- It can also be used to help security systems create security in a certain area.

The most recent version of YOLO is currently version 8 (Jocher et al., 2023), but there are many variations of this algorithm for various platforms, such mobile.

3.5.2 *PaddleDetection*

An open-source algorithm built on PaddlePaddle is called PaddleDetection. This approach provides real-time detection, multiple object tracking, instance segmentation, and object detection. More than 300 pre-trained models are provided by this algorithm, enabling the user to begin training with some data. Additionally, it is possible to train unique models. Also, it permits the usage of a variety of hardware platforms, including CPUs, GPUs, and TPUs. This method is being utilized in a variety of applications, including autonomous driving, visual inspections, medical picture analysis, security, and surveillance (Authors, 2019). It is a good choice because the algorithm is simple to use. Some advantages of this algorithm are listed below:

- **Varied library of functionalities** - Object identification, instance segmentation, key-point detection, and multi-object tracking are just a few of the object detection tasks that PaddleDetection's comprehensive model library can handle. This makes it an good choice for scientists and programmers who want to start using object detection right away.

- **Easy to start using the model**- PaddleDetection provides a well-documented API and is simple to use. Additionally, it offers numerous instructions and illustrations. Because of this, beginners who wish to learn about object detection should consider choosing it.

- **Flexible**- PaddleDetection is adaptable and may be made to fit the individual requirements of various users. Because of this, it's a great choice for programmers who wish to build customized object detection models.

3.5.3 *Detectron2*

The company Facebook AI developed the open-source object recognition framework Detectron2. It is among the most reliable and adaptable object identification frameworks available. Faster R-CNN, Mask R-CNN, and Point R-CNN are just a few of the object identification models that are supported by Detectron2. Additionally, it supports a number of training methods and data augmentation techniques. Furthermore, there is a sizable and active user and development community that can offer assistance and aid in debugging. Self-driving automobiles, visual inspection, and image analysis in medicine are just a few of the uses for Detectron2. It is a strong and adaptable instrument that may be utilized to address a variety of issues (Wu et al., 2019). This repository of ML models contains some of these features:

- **Accuracy-**One of the best object detection technologies available is Detectron2. It has been demonstrated that it performs better than other frameworks on a number of metrics, including COCO and PASCAL VOC.

- **Well-maintained**-Detectron2 is kept up-to-date and well-maintained. This implies that users can rely on the framework to be current and that any flaws will be quickly repaired.

- **Modular design-** Since Detectron2 is intended to be modular, it is simple to swap out the framework's various parts. This makes experimenting with various concepts and methodologies simple.

4

PROPOSED APPROACH

The primary objective of this master's thesis project is to identify the state of a municipal waste container, while concurrently enabling the identification of container types and the characterization of the waste proximate to them. Consequently, this chapter will demonstrate a workflow designed to realize the preceding objectives.

4.1 WORKFLOW

The system's performance will be heavily reliant on the performance of the machine learning prediction model. That is why having a strong dataset is critical. Three inputs from different sources will be considered before the data is prepared (there is no particular order of input):

- The initial input data originates from photographs of containers captured by the devices of a diverse group of individuals. These photographs contain a wide range of container types and diverse waste materials.
- The second data input comprises images sourced from websites like Google Maps or Mapillary, obtained through the utilization of the printscreen feature.
- The third input will be image data from a third-party mobile phone app that can indicate how full a municipal waste container is. This information can be used to determine whether or not the containers are full (bags of waste outside the correct location). Users of the application freely "feed" the application.

These three previously mentioned inputs will be critical in developing the fill prediction. After choosing the important data from these three inputs, which are depicted in the picture as Container Data, the data will be prepared by removing duplicates, missing data, and data that does not make sense for the situation at hand, among other things. Because of the volume of data, this stage is critical and can be time consuming.

The subsequent phase is anticipated to be also both resource-intensive and time-consuming. It comprehends the annotation of images, a pivotal step in model development. This

stage necessitates the formulation of a robust annotation methodology for delineating objects within the images, alongside the categorization of these objects into distinct classes. Furthermore, the selection of an appropriate annotation technology is imperative to ensure the accuracy and precision of the annotated data.

After the annotation procedure, a machine learning model must be selected to carry out the container fill level prediction. The study on the best algorithms for this particular topic, which was presented in chapter State of the Art (Chapter 3) will be used to guide the selection of this model.

The model will be trained after the algorithm has been chosen and implemented. This procedure will include a lot of trial and error, as the outcomes need to be carefully evaluated to ensure they align with the desired outcomes.

Finally, the prediction is ultimately product of the model predictor when utilizing the testing dataset. The ultimate aim of this process is to precisely distinguish the state of the container, including the status of its lid (open or closed) and the categorization of the container type.

The image 8 serves as a diagram, enabling the visualization of all that has been detailed previously. This, in turn, facilitates a more accessible analysis of the workflow.

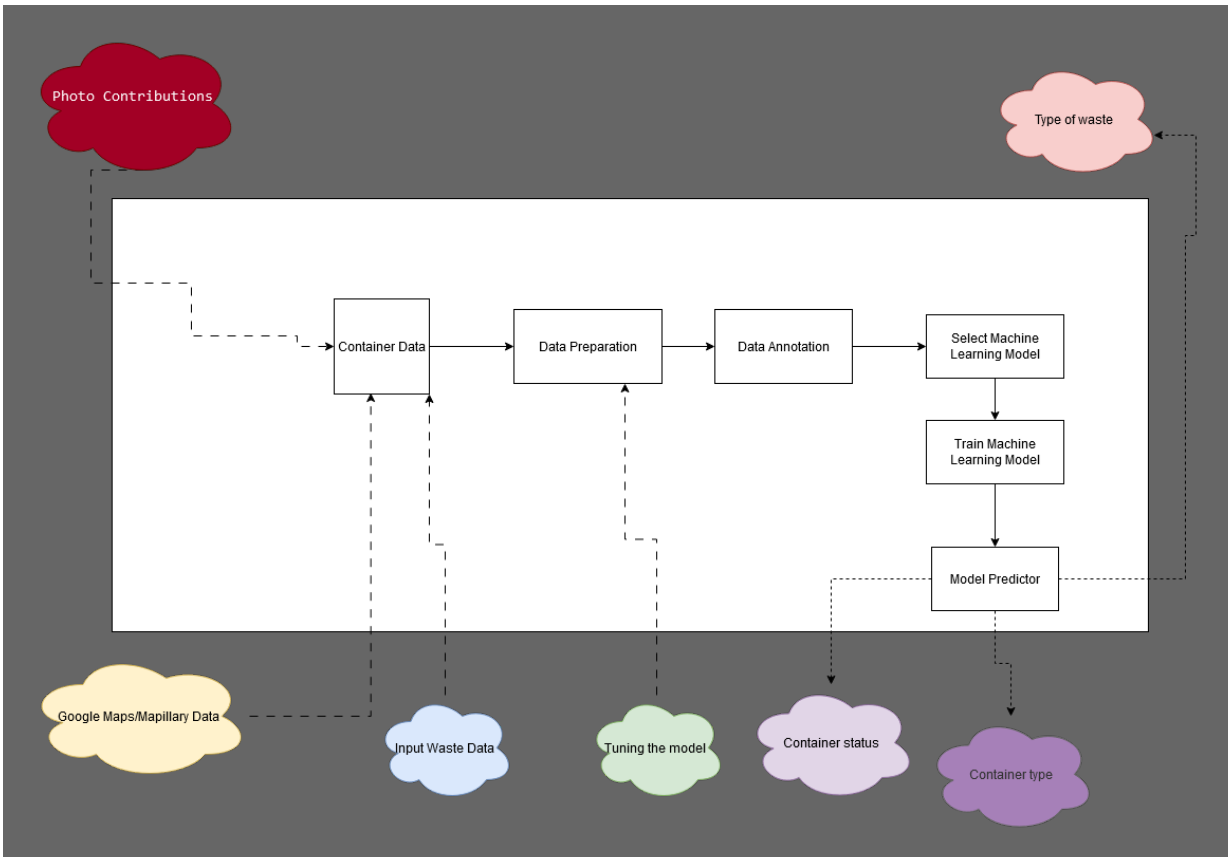


Figure 8: Workflow for predicting the state of a container, type of container and type of waste.

5

DEVELOPMENT

This chapter will go into the steps of development for this Master's project in great depth. It will also detail the problems encountered and decisions made during this process. The project development of this Master's project can be delineated into five principal categories:

- 1. Compilation and annotation of an initial dataset.
- 2. Training of a model to discern urban waste containers, encompassing all existing variants. This process facilitates the acquisition of images featuring at least one detectable container, thereby creating a novel dataset.
- 3. Formulation of a novel dataset through the inclusion of images in which urban waste containers have been identified.
- 4. Annotation of the images, demarcating distinctions based on the specific type of waste container.
- 5. Training the model with the final dataset.

It is also important to note that the model has always been trained locally, with personal hardware:

- **CPU-** AMD Ryzen 7 3800X 8-Core Processor 3.89 GHz
- **GPU-** NVIDIA GeForce RTX 2070 8 GB GDDR6
- **RAM-** 32GB (16GB + 16GB)

5.1 DATASET SELECTION

It is vital to collect a large amount of data to train the model in the future. In this master's project, it was anticipated that there would be three distinct sources of data:

- Data from a collection company of a certain municipality (e.g. collection time, collection location, container status, among others).

- Data from the user of the container
- Finally, frames from a third-party application that shows images of a certain municipality in real-time.

Several companies in the area that worked with waste collection were contacted throughout the production of this Master's project, including Braval (which deals with waste in Braga, Amares, and Póvoa do Lanhoso, among others), AGERE (which administers water and waste in Braga), and Resulima (manages waste collection in the municipality of Ponte de Lima), among others. All of these companies responded in the same way: no company keeps any record of the day/time of collection, the location of collection, the route traveled on the day of collection, the amount of rubbish collected, and so on. Typically, these companies gather rubbish in a highly antiquated manner. They walk past all of the waste bins, open them, and visually inspect whether or not they are full, before deciding whether or not to collect the garbage. This data search in garbage collection businesses thus proved fruitless. It was then decided to use ***Tidy City***, a third-party smartphone application that photographs and records numerous streets in a certain city inreal-time. It also aids in determining the status of a specific waste container in a particular area at a given time. This application has a vast database (more than 40000 pictures). These videos/images show a variety of everyday city objects, such as waste containers, graffiti, animals, and rubbish on the ground, among other infrastructures. Regarding this Master's project, the focus lies solely on videos and images containing representation of waste bins. This is to facilitate future detection of their fill levels by the state of their lid, container type, and adjacent waste. Then a new difficulty is introduced: identifying and separating the photos/videos in the collection that contain waste containers. Before developing a model to predict their fill levels by the state of their lid, container type, and adjacent waste, it was determined to create a model capable of determining whether a specific video/image contains a waste container. This pre-selection step enables the selection of images before employing the machine learning model to predict the state of the containers.

5.1.1 *Building the Initial Dataset*

This section will go over the decisions taken about data sources. Methods for collecting images for use in the picture pre-selection model.

CUSTOM DATASET FORMAT

The Yolov8 algorithm permits the use of custom datasets to train the model, but these datasets must be organized in a particular way. The technique being used determines the

directories order. Depending on whether someone wants to train the model using the bounding box, segmentation, or classification technique, the structure of these folders should look like what is described in the following in detail.

OBJECT DETECTION DATASETS OVERVIEW

In relation to the structure of the dataset employed for the detection of **bounding box** objects, it is pivotal to recognize that the dataset configuration file mandates the inclusion of a minimum of two folders: the train folder and the valid folder. The optional addition of a test folder is contingent on specific requirements. However, as elucidated subsequently, instances involving small datasets may stipulate the utilization of solely the train folder, rendering it the sole essential folder.

It is worth emphasizing that best practices advocate for the integration of all three folders, encompassing train, valid, and test folders.

These 3 folders must contain 3 different types of images and labels:

- **Folder *train*** - The majority of the images in the dataset are anticipated to be in this folder because this is where the images for training the model should go. Due to the initial tiny dataset (around 200-300 pictures) for this Master's project, it was decided to keep everything in the training folder so that could train with more examples and increase accuracy.
- **Folder *valid*** - The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters.This directory should not include images already present in the train folder. It should exclusively comprise images that the model has not been exposed to previously.
- **Folder *test*** - As previously mentioned, this directory is optional . Nonetheless, in practice, the assessment involved running tests on specific frames stored on the Tidy City server. The app's servers had around 40,000 frames, with container detections occurring in roughly 3,000 of them.

The organization of the directory can be seen in the tree in figure 9.

Figure 9: Custom dataset structure to train models using the bounding box technique, (Ultralytics, b)

The photographs and their labels must be contained in the folders named images and labels. The labels must share the same name as the image to be able to recognize them as belonging to that image. The labels should be in *.txt* format, while the photos might be in the widely used *jpg*, *png*, etc. formats. Additionally, the label *txt* file needs to follow a particular format where each line must contain the following information about an object instance:

```
<object-class> <x> <y> <width> <height>
```

- **Object's class index-** An integer is used to represent the class to which the object belongs (e.g. 0 for a person, 1 for an animal, etc.).
- **Coordinates of the object's center-** x and y correspond to the coordinates of the object's center, and are between 0 and 1
- **Height and width of the object -** the height and width of the object must also have a value between 0 and 1.

Here is an example of the YOLO dataset format for a single image with two object instances:

```
    0 0.5 0.4 0.3 0.6
    1 0.3 0.7 0.4 0.2
```

This structure and formats must be respected in the train, val and test folders. Additionally, the root folder must contain a file in the `.yaml` format (which acts as a configuration file). The dataset and model setup for training bounding box detection models are set up using this file. An example of how to create this `.yaml` file is shown below:

```
train: <path-to-training-images>
val: <path-to-validation-images>

nc: <number-of-classes>
names: [<class-1>, <class-2>, ..., <class-n>]
```

- The 'train' and 'val' fields specify the paths to the directories containing the training and validation images, respectively.
- The 'nc' field specifies the number of object classes in the dataset.
- The 'names' field is a list of the names of the object classes. The order of the names should match the order of the object class indices in the YOLO dataset files.

Either nc or names must be defined. Defining both is not mandatory. Here is an example of the `.yaml` file:

```
train: data/train/
val: data/val/

nc: 2
names: ['person', 'car']
```

INSTANCE SEGMENTATION DATASETS OVERVIEW

The folder structure and `.yaml` file structure are the same for a dataset prepared to employ the **segmentation** annotation technique, (Ultralytics, c), but the labels (`txt`) files format differs as follows:

```
<class-index> <x1> <y1> <x2> <y2> ... <xn> <yn>
```

- **Class index-** this integer value defines the class index for the object.
- **<x1> <y1> <x2> <y2> ... <xn> <yn> -** These are the segmentation masks of the object's coordinates. Spaces must be used to separate each coordinate.

Here is an example of the YOLO dataset format for a single image with two object instances:

```
0 0.6812 0.48541 0.67 0.4875 0.67656 0.487 0.675 0.489 0.66
1 0.5046 0.0 0.5015 0.004 0.4984 0.00416 0.4937 0.010 0.492 0.0104
```

The structure of the `.yaml` file is the same as above.

5.2 VERSIONS OF THE DATASET

5.2.1 *First version of Dataset*

Despite providing images containing waste containers, the Tidy City project encompasses a variety of infrastructure frames. Due to the complexity involved in manually selecting frames with containers from a vast pool of over 40,000 frames, a decision was made to initiate the process by increasing a substantial collection of waste container images. These images were sourced from personal captures and those of colleagues, spreading to locales such as Ponte de Lima, Braga, Barcelos, Valença, and Coimbra. The initial acquisition yielded around 200 images. Recognizing the insufficiency of this count, further images were procured using Google Maps and Mapillary, culminating in an aggregate of nearly 360 images.

5.2.2 *Latest Version of Dataset*

It was tested the performance of the YOLOV8 algorithm with the already-existing dataset (which contained roughly 360 photos). It quickly became apparent that the present dataset would need to be expanded. The results of the algorithm identified during the initial training effort were examined to expand this dataset, and it was determined that additional photos without intended objects were required. This is because these photographs were labeled as false positives by the algorithm. In Roboflow to annotate that the image has no object of interest, the null annotation feature is used. An example of a null annotation can be found in the figure 10. In this figure, there is a gate, which in some training sessions the model identified as a municipal waste container. By including this image in the dataset and marking it as null, the model will also learn what a container is not.

> *"A null annotation occurs when an image has no objects present, and thus, no bounding boxes need to be recorded. This is not necessarily problematic – in fact, it may be desired to train a model that objects are not always present in the frame."* (Nelson, 2020)

Figure 10: Picture annotated as null in Roboflow.

Subsequently, a collection of images was incorporated into the dataset, culminating in a total of **540 images** within the most recent version, denoted as version 19.

Note: It is possible to see all versions of the dataset in: containerdetectortrain (2023).

5.3 ANNOTATING THE DATASET

5.3.1 *Choosing Technique*

The initial annotation approach employed was the bounding box technique. This methodology also constituted the foundation for the preliminary training sessions.

However, as a means of conducting a comparative analysis between two distinct techniques to assess the potential enhancement in prediction accuracy while concurrently exploring alternative annotation tools, a strategic decision was made. This involved the dataset's deployment in two separate applications: Roboflow and Cvat.ia. Within the Roboflow platform, all images underwent annotation utilizing the instance segmentation technique. In turn, the Cvat.ia platform facilitated the annotation of all images through the bounding box technique. To showcase the differences in the layouts of the annotation applications, as

well as the variance in annotation techniques, one can observe an instance segmentation annotation in Roboflow and a bounding box annotation in Cvat in the images 11 and 12, respectively.

Figure 11: Annotation of picture using instance segmentation in Roboflow web app.

Figure 12: Annotation of picture using bounding box in CVAT.AI web app.

Note: A comprehensive comparison between these two tools can be found in Chapter 6, specifically in the Section 6.1.

5.4 TRAINING THE MODEL FOR PRE SELECT IMAGES

As previously said, this project collects frames from various infrastructures throughout the city. As a result, the majority of the frames are irrelevant to this Master's project. Consequently, a pre-selection of frames containing municipal waste containers is required. It was chosen to employ artificial intelligence to detect the presence of one or more waste containers in each frame in order to execute this pre-selection. YOLOV8, the most recent version of the algorithm, was selected. This version was chosen because it has a higher accuracy compared to others (StereoLabs), as well as a better integration with the Python language (Jocher et al., 2023), as it is now possible to install the library and use its functions ,something that was not possible in previous versions, in earlier versions, cloning the repository was a necessity to access the training scripts, detection features, and more.

Pre-training Configurations

It is crucial to confirm that the desired structure for the particular type of training is present during pre-training. The next step is to choose the location for the training; YOLOv8 supports training using either the command line or Python libraries. It was decided to train the models in the Yolov8 library for this project. In order to accomplish this, a notebook was made containing Yolov8 functions. Using the functions is rather straightforward, as shown in the following example:

```
from ultralytics import YOLO  # Import Library

model = YOLO('pretreinedmodel.pt') # Use pre-trained model

path=r"FILE PATH"
model.train(arguments) #train the model using the function 'train'
```

PRE-TRAINED MODEL

In order to speed up the training process, Yolov8 provides pre-trained training models (visible in table 3, are the pre-trained models for the desired annotation techniques), this helps the programmer as he starts training his model with some knowledge beforehand. In addition, these templates already contain a reasonable accuracy to get you started.

Model Type	Pre-trained Weights	Task
YOLOv8	yolov8n.pt, yolov8s.pt, yolov8m.pt, yolov8l.pt, yolov8x.pt	Detection (Bounding-Box)
YOLOv8-seg	yolov8n-seg.pt, yolov8s-seg.pt, yolov8m-seg.pt, yolov8l-seg.pt, yolov8x-seg.pt	Instance Segmentation

Table 3

There are five pre-trained weights provided by the model for each of the various tasks, and each of these five contains unique characteristics (as only pre-trained models were used for the bounding box and segmentation tasks, only these tables will be shown):

PRE-TRAINED MODEL - DETECT (BOUNDING-BOX)

Model	**size (pixels)**	**mAPval 50-95**	**Speed CPU ONNX (ms)**	**Speed A100 TensorRT (ms)**	**params (M)**	**FLOPs (B)**
YOLOV8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Table 4: Differences in Yolov8 pre-trained models (bounding box)(Jocher et al., 2023)

The information in table 4 was taken from the official documentation of the machine-learning model. The columns have the following meaning:

- **Model** - Name of the pre-trained model to be called as an argument.
- **size(pixels)**-model size in pixels.
- **mAPval (50-95)**-The average of the predictions, is a value that the higher it is, the higher the accuracy. How it is calculated can be seen in more detail, (Hui, 2018).
- **Speed CPU ONNX (ms)**- Importing and exporting ONNX models is straightforward in popular tools such as PyTorch and TensorFlow. For instance, the provided speeds (measured in milliseconds) represent the execution of the algorithm using pre-trained models in ONNX format on a CPU.

- **Speed A100 TensorRt (ms)** - TensorRT is a development kit developed by NVIDIA that allows developers to optimize inference using techniques such as quantization, layer and tensor fusion, kernel tuning, and others on NVIDIA GPUs. This column shows the execution time (in milliseconds) of the algorithm run by the GPU, using the pre-trained models.
- **params (M)**- Number of parameters used in each pre-trained model
- **FLOps (B)** - This is used to determine the performance of a computer, specifically in the field of scientific calculations, which make great use of floating point calculations; similar to instructions per second.It can be seen in more detail at Wikipedia

PRE-TRAINED MODEL - INSTANCE SEGMENTATION

The table 5 displays pre-trained models available for the instance segmentation technique.

Model	size (pixels)	mAPval 50-95	mAPmask 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n-seg	640	36.7	30.5	96.1	1.21	3.4	12.6
YOLOv8s-seg	640	44.6	36.8	155.7	1.47	11.8	42.6
YOLOv8m-seg	640	49.9	40.8	317.0	2.18	27.3	110.2
YOLOv8l-seg	640	52.3	42.6	572.4	2.79	46.0	220.5
YOLOv8x-seg	640	53.4	43.4	712.1	4.02	71.8	344.1

Table 5: Differences in Yolov8 pre-trained models for instance segmentation (Jocher et al., 2023)

- **Model** - Name of the pre-trained model to be called as an argument.
- **size(pixels)**-model size in pixels.
- **mAPval (50-95)**-The average of the predictions, is a value that the higher it is, the higher the accuracy. How it is calculated can be seen in more detail, (Krishnakumar, 2023).
- **mAPmask (50-95)**-The average of the masks (segments), is a value that the higher it is, the higher the accuracy. How it is calculated can be seen in more detail, (Hui, 2018).
- **Speed CPU ONNX (ms)**- Importing and exporting ONNX models is straightforward in popular tools such as PyTorch and TensorFlow. For instance, the provided speeds (measured in milliseconds) represent the execution of the algorithm using pre-trained models in ONNX format on a CPU.
- **Speed A100 TensorRt (ms)** - TensorRT is a development kit developed by NVIDIA that allows developers to optimize inference using techniques such as quantization,

layer and tensor fusion, kernel tuning, and others on NVIDIA GPUs. This column shows the execution time (in milliseconds) of the algorithm run by the GPU, using the pre-trained models.

- **params (M)**- Number of parameters used in each pre-trained model
- **FLOps (B)** - This metric is used to determine the performance of a computer, specifically in the field of scientific calculations, which make great use of floating point calculations; similar to instructions per second. It can be seen in more detail at (Wikipedia)

When it comes to choosing the most appropriate model for the project:

> *"The model size is linearly proportional to mAP and inversely proportional to inference time. Bigger models take more inference time to detect objects with higher mAP accurately. Smaller models have faster inference time but have comparatively lesser mAP. Bigger models are better if we have less data. Smaller models are more efficient if we have less space (edge scenarios)."* - (Krishnakumar, 2023)

To recognize recycling containers, the 'x' detection model was employed, **YOLOv8x**. The choice was made because the computer where the model will be trained initially has the processing power to train the "heavier" pre-trained model. Additionally, it has the best accuracy outcomes of any pre-trained weight. The dataset is quite limited, and as previously mentioned, it is prudent to employ larger models. Consequently, this will lead to longer training durations.

Used Hyperparameters For Performance Improvement

Throughout the training process, efforts were made to enhance performance, specifically in terms of reducing training time, using various function Hyperparameters. All Hyperparameters and their corresponding functions are detailed in the documentation (Ultralytics, a). This subsection describes the hyperparameters used and their value when training the model. Although the procedure takes a while by itself, it can be sped up by making a few changes:

CACHE

This parameter governs the decision to cache or refrain from caching the image dataset in the system's RAM. This choice carries the potential to enhance the operational efficiency of YOLOv8, even if at the expense of consuming a substantial amount of system memory.

The default configuration for this parameter is set to None, indicating that YOLOv8 does not engage in caching the image dataset as a standard practice. Notably, the necessity for dataset caching depend upon the dimensions of the image dataset in use. Smaller image datasets may not ensure the utilization of caching, although larger counterparts could benefit from its implementation, leading to improved performance of YOLOv8. The table 6 shows all the parameters and their effects on the cache:

Value	Effect
None	Do not cache the image dataset.
True	Cache the entire image dataset to RAM.
ram	Cache only the images that are currently being processed to RAM.

Table 6: Values of the cache parameter and their effects.

Given the relatively small dataset, the cache value remained at its default setting, which is "None."

BATCH

The batch value in YOLOv8 controls the number of images that are processed at the same time. Higher batch value can improve the performance of YOLOv8, but it can also use more memory. The default value of batch for training session's is **16**. When employing a system equipped with abundant memory resources, it becomes plausible to increase the batch size as a means to enhance the operational efficiency of YOLOv8. On the other hand, when operating within the bounds of a memory-constrained environment, it becomes necessary to decrease the batch value to avert the risk of memory reduction. Yolov8 also offers the convenience of an automatic batch value, allowing the model to adjust to the current hardware constraints during training. This can be achieved simply by setting the batch value to -1. In the specific case of this master's project, the batch value used was -1, where the model automatically detected the best possible value for the hardware constraints.

DEVICE

This parameter within the context of YOLOv8 delineates the designated computational device for the execution of either training or inference processes. The default configuration for the "device" parameter is set to CPU, signifying YOLOv8's utilization of the central processing unit for said purposes. However, in instances where a graphical processing unit

(GPU) is at one's disposal, the "device" parameter can be properly employed to explicitly designate the preferred GPU for YOLOv8's operation.

- **cpu -** use the CPU.
- **0 (cuda) -** use GPU 0.
- **n (cuda) -** use GPU n.

In this training session, the GPU was employed as the chosen device. This decision was based on research (Pysource) that indicated it as an effective means to accelerate the model's training speed.

CUDA

CUDA stands for a parallel computing structure coupled with an application programming interface (API) that has been developed by Nvidia. This framework facilitates the utilization of specific categories of GPUs for the purpose of general computing tasks and exhibits a computational paradigm referred to as general-purpose computing on GPUs (GPGPU). A variety of applications can be accelerated using the potent CUDA tool. It is particularly well suited for computationally demanding applications like machine learning, computer vision, and scientific computing (OH, 2012) (git, 2022). Benefits of CUDA in the creation of this master project include:

- **Performance-** CUDA can return substantial enhancements in the performance of applications characterized by computationally demanding workloads. This efficacy is attributed to the inherent quickness of GPUs, which outpaces CPUs notably in the domain of parallel computational assignments.
- **Scalability-** CUDA exhibits the capacity for scalability to accommodate larger and increasingly complex applications. This phenomenon emerge from the ability to increase the computational potential of a system by incorporating additional GPUs.
- **Portability-** CUDA code is portable across diverse platforms, including operating systems such as Windows, Linux, and macOS. This attribute makes CUDA a prudent selection for applications necessitating deployment across a vast spectrum of systems.

SAVE PERIOD

This parameter within the YOLOv8 framework governs the interval at which the model's weights are saved during the training process. **By default, the save period value is set to -1, that is, any value smaller than 1 is not saved by the model during the training process.**.

Nevertheless, it is within the choice of the user to modify the save period value to either increase or decrease the frequency of model stored.

In scenarios involving extensive training durations, the possibility of increasing the frequency of weight stored may be advantageous. Such an approach serves to mitigate potential adversities that could appear during the training process, enhancing the chance of recovery. However, it is important to acknowledge that increasing the frequency of weight stored will concurrently necessitate a proportional expansion of disk space allocation, and in the case of long training sessions, this can be a problem.

learning rate (lr0 and lrf)

The YOLOv8 framework incorporates the lr0 and lrf parameters, which collectively manage the model's learning rate throughout the training process. The lr0 parameter denotes the initial learning rate, while the lrf parameter refers to the final learning rate. This learning rate essentially regulates the pace at which the model's parameters undergo adjustments during the training process.

The default settings for lr0 and lrf are established at 0.01 and 0.001, respectively. This signifies an initial learning rate of 0.01, and 0.001 is the final learning rate at the last epoch of training. Nonetheless, users retain the liberty to customize these values according to their preferences.

In the training scenarios, a prudent strategy involves commencing with a reasonable initial learning rate, such as 0.001, to anticipate the occurrence of overfitting tendencies on the training data. Subsequently, the learning rate can be incrementally increased as the model assimilates information.

Conversely, when confronted with time-constrained training undertakings, opting for a higher initial learning rate, such as 0.01, can accelerate the learning process. It is imperative to note that a reduction in the learning rate over the course of training becomes important to mitigate overfitting occurrences and align the model's performance with the training data.

In the different training tests carried out, changing this value did not show any improvement, so it was decided to keep its original value.

epochs

This hyperparameter instructs the model regarding the number of epochs it should undergo during training. **By default, the epochs parameter is 100.** It is, however, within the choice of the user to decide the epochs value to any desired number.

The specific value of this hyperparameter should be adjusted according to the desired level of accuracy. Subsequently, a process of iterative experimentation can be launched,

where varying values of epochs are evaluated to determine the optimal configuration that returns the most favorable outcomes for the given application.

Several considerations must be taken into account when determining the configuration of the epochs parameter:

- Normally, an increase in epoch count during model training is correlated with an improvement in model accuracy.
- Training a model using a restricted dataset necessitates a proportional increase in the epoch count to obtain the targeted accuracy threshold ,however, it can lead to overfitting, which is why the state of the model must be monitored during the training process.
- In the case of a robust dataset, achieving a satisfactory level of accuracy within a diminished epoch count is feasible.

Yolov8 training function example in Python with some of the above-mentioned hyperparameters:

```
model.train(data='path/to/dataset.yaml',epochs=400,device=1,save_period=20)
```

This means that the model will train 400 epochs, through Gpu(device=1) and save the model every 20 epochs.

5.4.1 *Training the model to pre-select frames*

Although a total of 96 training sessions were conducted to refine the model, a considerable portion of these iterations was obsolete due to adjustments made to hyperparameters. Notable hyperparameter modifications encompassed parameters such as device selection, save period, among others. The objective of these adjustments refer to the optimization of the training process.

Throughout the course of these training iterations, a meticulous analysis of the results was undertaken. This examination encompassed the assessment of two key aspects: firstly, the continuation of learning progress, as evidenced by a diminishing "loss" metric and a concurrent increase in accuracy; and secondly, the surveillance for indications of overfitting.

To execute this strategy, an initial modus operandi was established where the model underwent training every 100 epochs. During each training iteration, the previously generated best model was selected as a benchmark for comparison and refinement.

The initial training session entailed the utilization of the dataset annotated with **bounding box technique**. Opting for the pre-trained **Yolov8x model** was a deliberate choice, due to its good accuracy outcomes, despite the extended training duration it necessitates. As previously highlighted, when dealing with limited datasets, the adoption of more resource-intensive models is a preferred approach, even if this results in extended training periods. During the subsequent training session, the dataset annotated via the **instance segmentation technique** was employed. Similarly, the selection of the pre-trained model refer to **YOLOv8x-seg**, which, although resource-intensive, aligns with the pursuit of higher accuracy.

The final training phase encompassed a total of 350 epochs for each dataset, ultimately yielding the most favorable outcomes in regard to loss metrics and accuracy. Subsequently, a thorough analysis and comparison were conducted to find out the most suitable method and model for detect urban waste containers from the frames provided by the application.This training lasted about 2800 minutes (47 hours).

Note: The Results along with the selected model can be observed within Chapter 6, precisely within the Section 6.2.

5.5 FORMULATION OF A NOVEL DATASET

5.5.1 *First Version's of the new dataset*

Following the model selection process, an inference was conducted within the Tidy City application database to pre-select images, resulting in the identification of approximately 2798 images featuring urban waste containers. Subsequent analysis of these images revealed the presence of certain false positives, which were promptly identified and removed.

Moreover, as these images were sourced from video frames, instances occurred where the recording device remained stationary for an extended period, resulting in the continuous capture of the same container. In some cases, as many as 100 frames were extracted from a single container. To mitigate the risk of overfitting in the final model, these redundant images were systematically eliminated.

To improve the dataset further and diversify the representation of litter bins, semi-underground, and underground containers, additional images were sourced from Google Maps. As a result of these efforts, the first final version of the dataset comprised approximately 1000 images, ensuring a balanced and comprehensive dataset for model training and evaluation.

Furthermore, within this dataset, specific images have been designated to serve as validation images. These images are deliberately excluded from the training dataset, giving new

stimuli to the model. Their inclusion enables a more authentic evaluation of the model's metrics, facilitating a realistic assessment of its performance.

The decision to employ Roboflow for annotating these images was made due to its ease of use and the convenience it offered for exporting the results efficiently.

5.5.2 *Final version of the new dataset*

Throughout the training process for this new model, it became clear how important it was to have even more images in the dataset. Therefore, a new version was generated which contained the following data augmentations:

- **Flip Horizontaly**- In this updated dataset, each image includes its original version and an additional duplicate that has been horizontally rotated.
- **Greeyscale Filter**- To enhance the model's capability in discerning the pixel maps of individual objects, a gray filter was introduced. Essentially, this filter aimed to facilitate the identification of container types (e.g., 2-wheel, 4-wheel) based on their shapes. It's noteworthy that this filter was applied to only 25 percent of the images within the dataset.

After implementing these data augmentations, the final count of images in the dataset was of **1960** images.

Note: This and other versions of this dataset are public and can be viewed at: stateofwastebins (2023).

5.6 Annotation of the Final Dataset

The second stage of annotation proved to be considerably more complex. This complexity came from the need to establish distinct classes for the categorization of **container types**, **container functions**, **container conditions (open, closed, or full)**, and **waste types**. It is imperative to highlight that the annotation process was exclusively conducted using the **instance segmentation method**. This approach was chosen due to the necessity for a detailed object analysis within the image, particularly when assessing the container's condition. Through this method, it becomes feasible to construct a pixel map for the object of interest in the image, striving to establish discernible patterns indicative of whether the container is open, closed, or full. To establish a well-structured framework for the annotation process, a hierarchical approach was employed. This methodology led to the

creation of two distinct trees. In the first tree, annotated containers were systematically organized into a hierarchical structure, with all containers originating from a primary class designated as **waste bin**. Subsequently, the annotation tree branched out, classifying containers according to their specific types, facilitating a methodical and precise annotation process. Simultaneously, the second tree, addressing the types of waste typically found adjacent to these containers, commenced from a primary branch labeled as **waste** and further branched out to delineate various waste categories. To make this description easier to understand, the first branch of the tree can be seen in the figure 13.

Note: The outcomes along with the selected model can be observed within Chapter 6, precisely within the Section 6.2.

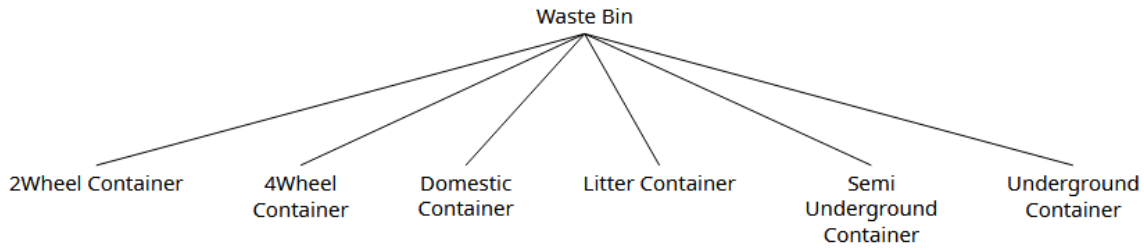


Figure 13: First branch of the waste containers class tree.

Together with the waste container hierarchy, an additional tree structure was developed specifically for the identification of waste types.

5.6.1 *2 Wheel Container*

This particular container class encompasses containers characterized by their possession of two wheels and may be distinguished by:

- **Waste type by color** - Blue, yellow, green (designating glass), normal for undifferentiated waste.
- **Container's condition** - Open, closed, or full (indicating overfill).

In the image 14, a exemplar of this container type is observable:

Figure 14: Real examples of 2 wheel waste containers

All the classes described in image 15 derive from a main class known as **2wheel** which, in turn, inherits from the superclass **waste bin**, in figure 13.

Note: The colors next to the classes are the colors associated with the containers, and are only there to make it easier for the reader to analyze the tree.

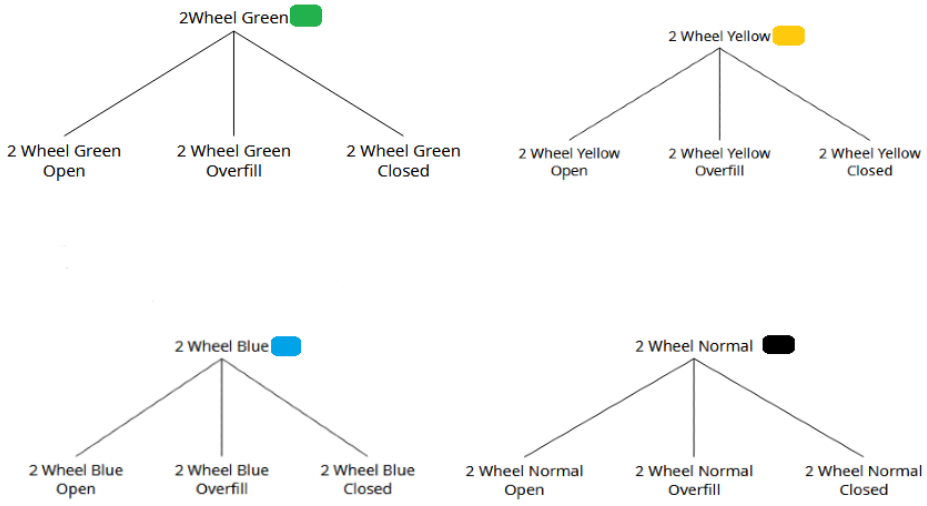


Figure 15: 2 Wheel Containers Branches.

5.6.2 *4 Wheel Container*

This particular container class incorporate containers characterized by their possession of four wheels and may be distinguished by:

- **Waste type by color** - Blue, yellow, green (designating glass), normal for undifferentiated waste.
- **Container's condition** - Open, closed, or full (indicating overfill).

In the image 16, a exemplar of this container type is observable:

Figure 16: Real examples of 4 wheel waste containers

All the classes described in image 17 derive from a main class known as **4wheel** which, in turn, inherits from the superclass **waste bin**, in figure 13.

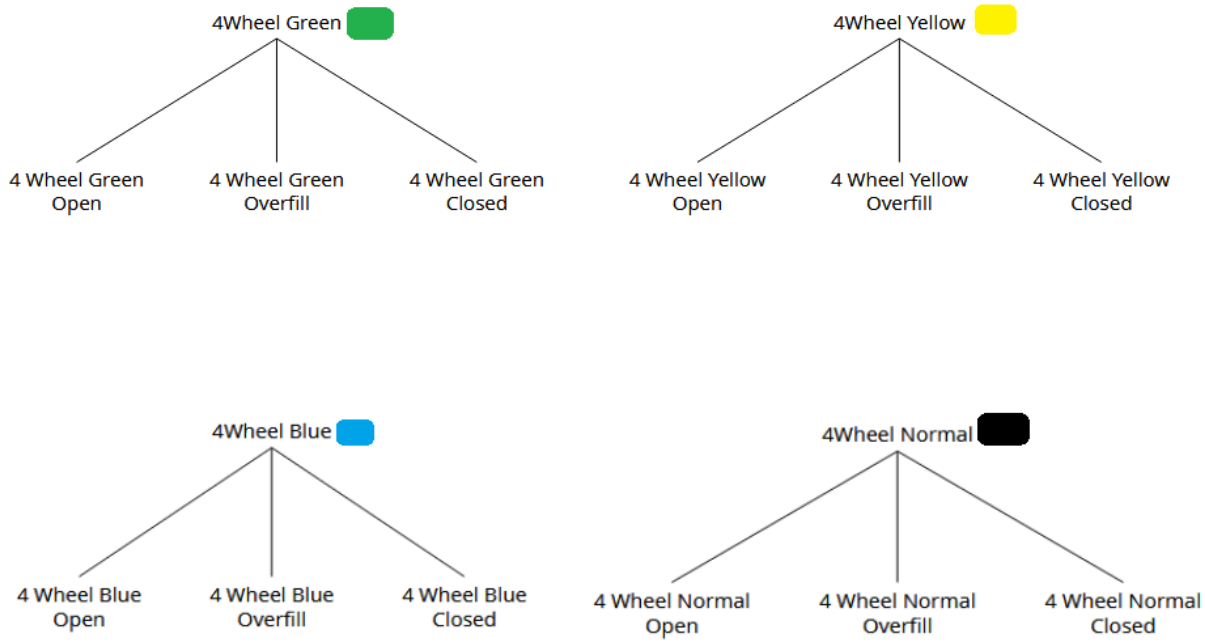


Figure 17: 4 Wheel Container's Branches.

5.6.3 *Underground Waste Containers*

Containers of this variety are firmly affixed to the ground and encompass substantial reservoirs for waste allocation. Typically situated within ecological islands, they house receptacles designated for glass, plastic, and paper materials, in addition to a section for unsorted waste.

- **Waste type by color** - Blue, yellow, green (designating glass), normal for undifferentiated waste.
- **Container's condition** - Open, closed, or full (indicating overfill).

In the image 18, a exemplar of this container type is observable:

Figure 18: Real examples of Underground waste containers

All the classes described in image 19 derive from a main class known as **Underground Container** which, in turn, inherits from the superclass **waste bin**, in figure 13.

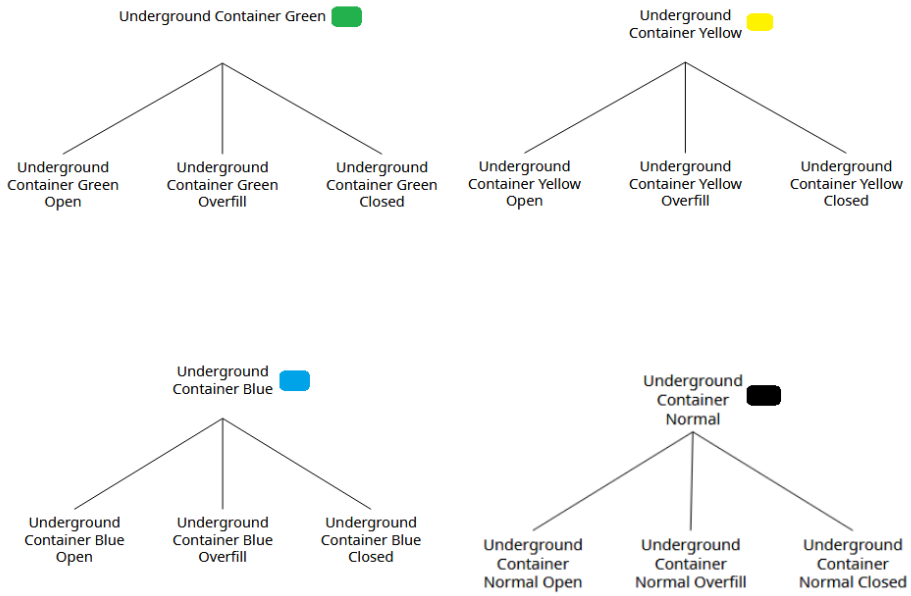


Figure 19: Underground Container's Branches.

5.6.4 *Semi Underground Waste Containers*

These containers differ from the last type in several ways.These bins, instead of having the waste container underground, are normal bins without wheels and with the waste container visible. Notably, these containers maintain a single condition, which is "overfill," as they are perpetually in an "open" state and lack a lid.

- **Waste type by color** - Blue, yellow, green (designating glass), normal for undifferentiated waste.
- **Container's condition** - Full (indicating overfill).

In the image 20, a exemplar of this container type is observable:

Figure 20: Real examples of Semi Underground waste containers

All the classes described in image 21 derive from a main class known as **Semi Underground Container** which, in turn, inherits from the superclass **waste bin**, in figure 13.

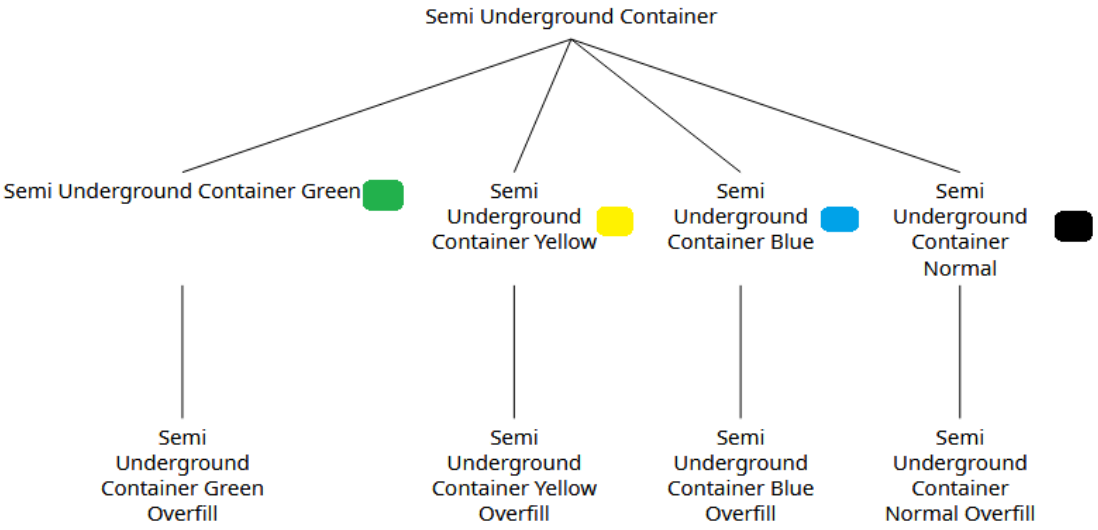


Figure 21: Semi Underground Container's Tree.

5.6.5 *Domestic Bin*

These container types are compact bins allocated to residents, commonly placed along public roads for collection by urban waste management crews. In this instance, container differentiation is not based on color; the primary consideration is whether the container is full or not.

- **Container's condition** - Full (indicating overfill).

In the image 22, a exemplar of this container type is observable:

Figure 22: Real examples of Domestic Bins

All the classes described in image 23 derive from a main class known as **Domestic Bin** which, in turn, inherits from the superclass **waste bin**, in figure 13.

Figure 23: Domestic Bin Tree.

5.6.6 *Litter Bin*

The so-called litter bins, positioned throughout the municipalities and usually attached to some structure, such as a bus stop, lamppost, among others. These containers are smaller in volume and only contain one state, overfill.

- **Container's condition** - Full (indicating overfill).

In the image 24, a exemplar of this container type is observable:

Figure 24: Real examples of litter bins

All the classes described in image 25 derive from a main class known as **Litter Bin** which, in turn, inherits from the superclass **waste bin**, in figure 13.

Figure 25: Litter Bin Tree.

In the figure 26, representing a hierarchical tree, one can observe a comprehensive array of waste types that the model will undergo training to identify.

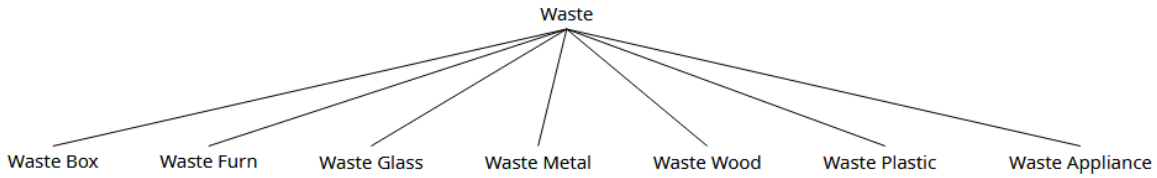


Figure 26: Waste Tree.

Each subordinate class within this hierarchy corresponds to a specific type of waste, as elucidated below:

- **Waste Box** - This class encompasses all forms of paper waste, including items such as paper boxes, sheets of paper, paper bags, and similar materials.
- **Waste Furn**-This class refer to all waste categorized as furniture, encompassing items like chairs, bed frames, cupboards, and similar articles. Frequently, this subclass coexists with other subclasses such as Waste Wood, Waste Metal, Waste Glass, Waste Plastic since items like wooden chairs fall under the category of furniture and are constructed from wood.
- **Waste Glass**- This class encompasses all types of waste comprised of glass materials, such as glass table tops, glassware, mirrors, and similar items.
- **Waste Metal** - This class encompasses all objects constructed from metal materials, including items like bed frames, chair legs, table legs, and other metal-based articles.
- **Waste Wood**-This class encompasses all objects crafted from wood materials, encompassing items like furniture, chairs, pallets, and various other wooden objects.
- **Waste Plastic**-This class incorporates all objects crafted from plastic materials, encompassing items like toys, plastic bags, packaging, and other plastic-based items. It is often combined with other classes due to its abundant presence in waste materials.
- **Waste Appliance**- This class encompasses all household appliances, including items such as dishwashers, washing machines, hairdryers, and various other domestic devices.

5.7 CONFIGURATION OF THE FINAL TRAINING SESSIONS

The training process for this concluding phase of the Master's project closely resembled that of the previous iterations. Initially, tests were executed on the dataset, and the outcomes were subjected to comprehensive analysis. It was deduced that an augmentation of the dataset was imperative. Subsequently, in all subsequent training sessions, the best-performing model

from the preceding session was employed, thus ensuring a coherent progression of the training process.

5.7.1 *Initial Phase of Training Sessions*

The training methodology closely resembled that of previous iterations. The model underwent training for a total of 20 epochs. The initial training sessions, typically ranging 5-10 epochs, involved a limited number of epochs (around 20-50) and were primarily aimed at determining optimal values for specific hyperparameters, including batch and RAM cache, among others.

Following this initial phase, the model underwent a warm-up phase, also comprising a few epochs, approximately 10. Subsequently, the actual experimental training sessions commenced.

Throughout the training regimen, the model's learning progress was meticulously scrutinized. At intervals of every 100 epochs, the state of the best-trained model was assessed. As long as no overfitting was observed, the model continued to be trained until an optimal model state was achieved.

In contrast to previous training processes, there is a notable difference in this particular training process. The dataset was partitioned into two distinct folders, each serving a distinct purpose. The training folder, housing the majority of the images, was employed for model training. On the other hand, the validation folder was populated with images that were absent from the training folder, thereby constituting images that the model had not been exposed to previously. The selection of images for the validation folder was carried out with consideration for the diversity of container types, container conditions, and the adjacent waste.

In the training process, the YOLO algorithm assesses the model's performance after each epoch or at predefined intervals using the validation dataset. The model's predictions on the validation data are joined with the ground truth, and evaluation metrics such as mean average precision (mAP) are computed to gauge its accuracy and localization performance.

The initial phase involved conducting a series of test runs to ensure the state of the model for training. Subsequently, the actual training process commenced. The training configurations were as follows:

- **Pre trained model** - The pre-training model selected was **YOLOv8x-seg**. This choice aligns with previous training sessions, as it exhibit superior accuracy, even if at the cost of a longer training duration. Given the relatively smaller dataset for this training session, it proved to be the most suitable option.

- **Total of Epochs**-A total of **400 epochs** were configured for training in this instance. The decision to raise the number of epochs in comparison to prior training sessions was based on the expanded dataset size and the greater number of classes to be accommodated.
- **Batch** -Based on insights gathered from prior training sessions, it was determined that the optimal batch value could only be set to **6**. Hardware constraints precluded the possibility of increasing this value.
- **Cache** - To accelerate the training process, the RAM cache was activated in an attempt to minimize the duration required for training completion.
- **Save Period** -For continuous monitoring of metrics during the training process, the model was configured with a save period value of **50**. This setting ensured that the model saved the best result achieved at intervals of every 50 epochs.

It's important to highlight that these initial training sessions were conducted using the early versions of the most recent dataset. Consequently, at the beginning, the number of images in these training sessions was approximately 1000.

Note: The outcomes of this initial phase training can be referenced on Chapter 6, specifically within the Section 6.3, and its corresponding sub-section,Section 6.3.1.

5.7.2 *Final Phase of Training Sessions*

During final phase of this training phase, the configuration settings remained consistent with the preceding one. However, there was a notable alteration in the approach. Instead of utilizing a pre-trained YOLO model, **the model employed was the best-performing version from the last training session of the initial phase**. Furthermore, the latest iteration of the dataset was employed, featuring augmentations, resulting in a total of approximately 1900 images.

In this second and final phase, approximately 1,000 training sessions were scheduled. Each session was interrupted every 100 epochs for analysis. During these interruptions, the current state of the model was assessed, including its learning progress and loss metrics. If the model demonstrated satisfactory progress, the training process continued using the best model from the previous session. However, it was not possible to reach 1,000 epochs because the model showed no improvement beyond epoch 650. Therefore, the default patience

parameter of 50, which determines when to stop training if no improvement is observed in 50 epochs, was applied.

Note: The outcomes of this initial phase training can be referenced on Chapter 6, specifically within the Section 6.3 and its corresponding sub-section,Section 6.3.2.

5.8 Brief study on another algorithm

Given the unsatisfactory results obtained after 650 training epochs with the YOLOV8 algorithm, an alternative approach was adopted for training the latest version of the dataset, which included around 1900 images. The alternative chosen was to use the Faster R-CNN model with the ResNet+FPN backbone, (Ren et al., 2015) via the **Detectron2** repository.

The Detectron2 repository contains models that are remarkably capable of achieving greater precision compared to YOLOV8. However, it has the disadvantage of requiring more computational resources and longer training times. In summary, Detectron2 models stand out for their accuracy, while YOLOv8 is recognized for its speed and efficiency. The choice between these models depends on the specific requirements and constraints of the task at hand.

Although these models offer advantages in terms of accuracy, they consume more computational resources and require longer training times. Due to time constraints within the scope of this master's project, a more in-depth exploration of this algorithm was a challenge. However, even with a somewhat limited exploration, the notable advantages of this algorithm compared to YOLOv8 became evident:

- **Greater variety of pre-trained models** -Detectron2 provides a diverse selection of pre-trained models, and the choice of which model to utilize should align with specific requirements and system constraints. It is possible to check all the pre trained models in , (Research).
- **Better Accuracy**- If high accuracy is a requirement and there are abundant computational resources available for training and deploying complex models, Detectron2 may be the preferred choice.

  On the other hand, if real-time inference speed is a top priority and a minor reduction in accuracy compared to high-end models is acceptable, YOLOv8 or similar YOLO variants might be more suitable (Shashank, 2021).
- **Best suited to segmentation tasks** - While YOLOv8 can be adapted for segmentation tasks, its primary design focus lies in object detection. Consequently, its architecture

may not be as ideally suited for complex pixel-level segmentation tasks when compared to the dedicated models provided by Detectron2.

6

RESULTS AND ANALYSIS

Within this chapter, analysis of the results accumulated throughout the progression of this master's thesis will be presented and expounded upon.

6.1 COMPARISON BETWEEN CVAT.IA AND ROBOFLOW

Both software contain advantages and disadvantages to their use, and how they were applied gave rise to the possibility of identifying their differences. Both are open source and free to use, however to access some of the more advanced features a subscription is required.

6.1.1 *Roboflow*

Regarding Roboflow, the following features were identified as an advantage:

- **Export in Yolov8 format** - Roboflow makes it simple to use the dataset in python scripts by allowing the user to download the dataset in several versions of YOLO (as well as other formats) and further organizing the folder structure into the structure that YOLO demands. Additionally, while downloading, directories containing photos and txt files with annotations are included.
- **Easy Integration with Python**- This service has the benefit of including python libraries, which enables programmers to utilize functions to download the dataset more automatically, as demonstrated in the sample below created for use with Jupyter notebooks.

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="APIKEY")
project = rf.workspace("gdit").project("aerial-airport")
dataset = project.version(1).download("yolov8")
```

- **Numerous public datasets** - More than 100,000 publicly available datasets with annotations are available to train models using a variety of techniques, including segmentation, classification, and bounding box. This enables the model's accuracy to increase by training it with additional photographs.
- **Segment Anything Model (SAM)**- The platform offers a function that enables the annotator to reduce time while maintaining the accuracy of the annotation while using the segmentation annotation approach. SAM segments things using cloud artificial intelligence. This tool's name is **Smart Polygon** , (Lynn).
- **Generate Augmentations**- Furthermore, in an effort to expand the dataset and enhance the training of the model, the data augmentation functionality provided by Roboflow was employed.

  Data augmentation enables the transformation of the dataset through the manipulation of existing images. This tool can apply various modifications, including vertical and horizontal rotations, as well as the addition of filters such as grayscale and color inversion, among others. Additionally, users have the flexibility to specify the percentage of images to undergo transformation. In aggregate, this feature significantly contributes to improving model accuracy. For a more detailed exploration of this feature, please refer to the following link: Roboflow.

However, during the course of this Master's project there were some problems with this platform:

- **Class Creation** - When creating classes, classes must be created on each image that is to be annotated. This means that the user could not create them before the annotation, so after the annotation, the user could choose the same one. This makes the annotation more prone to typos.

6.1.2 *Cvat.ia*

As far as Cvat.ia is concerned, the advantages are as follows:

- **Class Creation**- In this web application, the classes must be created before the annotation; after that, all that is left to do is select the class to which the object belongs. Thus, typing errors are reduced.
- **Keyboard Shortcuts** - Although Roboflow featured certain shortcuts as well, they were more user-friendly and made it quicker to annotate images in this application.

However, during the course of this Master's project there were some problems with this platform:

- **Export annotations** - YOLOV8 is one of the formats in which the made annotations can be sent, although in the tool's free edition, one can only export the annotations—not the images. Additionally, the tool frequently produced mistakes when exporting segmentation-based annotations; to overcome this, the annotations were exported in COCO JSON format and then translated to YOLO format using a script.
- **Identified as unsafe** - On several occasions, upon accessing the annotation platform, the web browser encountered an error image indicating that the URL was deemed insecure. This issue manifested across various computers and access points, displaying a consistent occurrence.

6.2 Results of training the model to pre-select images

This section will comprehensively document and analyze the outcomes derived from multiple training iterations of the model, which was specifically trained for the task of pre-selecting images from the Tidy City application dataset.

Note: For a more in-depth understanding of pre training process, detailed information is in Chapter 5, Section 5.4.

6.2.1 *Results of the training with Bounding Box Dataset*

The subsequent sections will elucidate and comment upon the outcomes of the training procedure utilizing the dataset made for municipal waste container identification via bounding boxes. This training endeavor aimed to preselect frames within the Tidy City database and will be comprehensively detailed and discussed. The metrics depicted in the graph in the figure 27 are briefly elucidated below.

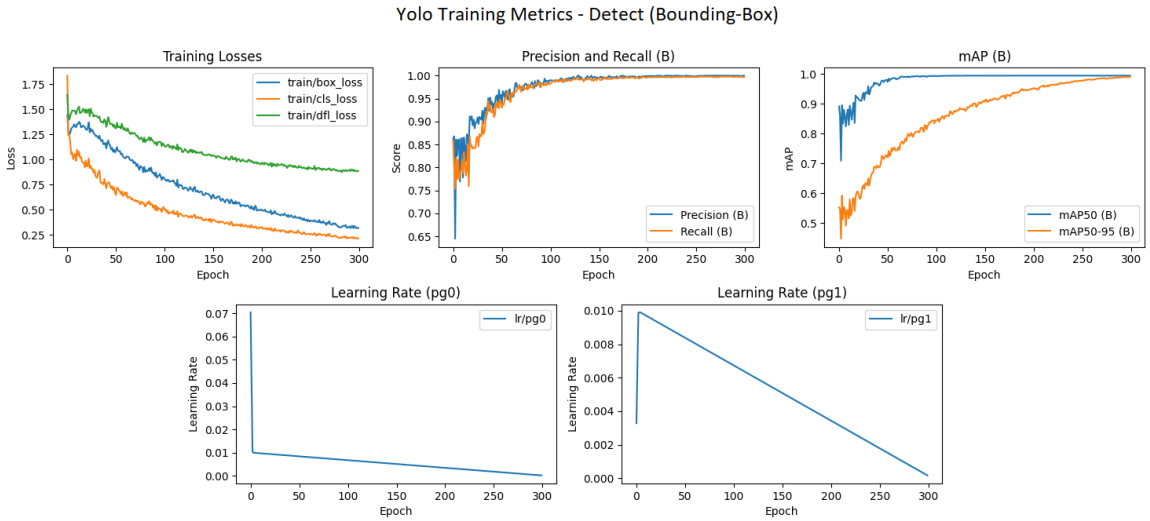


Figure 27: Best Results of the train with bounding box dataset

Box Loss

The term "training box loss curve" denotes the graphical representation of loss values throughout the training procedure. Within the YOLO framework, this curve typically showcases the diminishing trajectory of the loss function over consecutive training iterations or epochs. This visual representation provides insight into the model's learning in accurately and efficiently predicting bounding boxes encompassing objects.

The loss function serves as a metric to quantify the dissimilarity between predicted bounding boxes and the actual ground truth bounding boxes. During the training regimen, the model endeavors to minimize this loss by iteratively adjusting its parameters, which encompass weights and biases, based on the gradient of the loss function. As the training unfolds, the ideal trajectory involves a gradual reduction in loss, indicative of the model's progressive improvement in its capacity to make accurate bounding box predictions.

As discernible from the inaugural graph (the leftmost plot, **blue line**) in figure 27, the curve manifests a progressive decline over the temporal course, signifying the model's progression in mastering the task of accurately predicting the bounding boxes cover objects within the images.

Cls Loss

The abbreviation "Cls" corresponds to the term "classification." In the dimension of object detection, the model is responsible for not only predicting bounding boxes around objects but also assigning class labels to the identified entities. The classification loss metric measures how effectively the trained model can segment the original image into smaller regions, select (localize) the most salient of these regions, and classify any detected objects

(if present). As discernible from the figure 27 (leftmost plot, **orange line**), the preliminary loss was notably elevated, as anticipated given the model's lack of familiarity with waste containers. Subsequently, a gradual reduction in loss is evident, which is indicative of a favorable trend. This diminution signifies the model's progression in acquiring an enhanced aptitude in accurately predict the class attributions of objects depicted within the images.

Dfl Loss

The acronym "DFL" corresponds to Distribution Focal Loss, which takes into account the class imbalance during training. This imbalance appear when one class is more prevalent than another. For instance, in an image of a street where the goal is to detect both cars and bicycles, out of 100 photos, one might have 200 objects classified as cars and only 10 objects classified as bicycles. Due to this disparity in object counts within the image, the model tends to learn to detect cars more accurately than bicycles. Consequently, whenever the model attempts to classify an object as a bicycle, the loss is augmented.

The pertinence of maintaining a low value for this metric is underscored, signifying the accurate understanding of objects within the images and their corresponding spatial coordinates. Evident from the visualization of the figure 27 (leftmost plot **green line)**, the observed decline in this curve over the course of time signifies the model's progressive improvement in its capacity to accurately predict bounding box attributes, thus reinforcing the effectiveness of its learning process.

Precision (B)

The metric known as "Precision (B)" is reflective of the accuracy of the bounding boxes acquired during the model's training process. Precision, in a large sense, indicates the precision of positive predictions made by the model. It is determined by dividing the count of true positives by the sum of true positives and false positives.

In the present context, this metric serves as an indicator of how many objects detected by the model are indeed relevant.

The depicted curve in figure 27, illustrates the model's attained accuracy across varying epochs of training progression. Commencing at a lower point, the curve exhibits an ascending trend over time, indicative of positive advancement. This progression signifies the model's evolving competence in achieving more precise predictions of object bounding boxes within the images.

The curve culminates in a peak accuracy of approximately 0.9, reached at epoch 200. This substantial accuracy benchmark attests to the model's capacity to discern objects within the images with a some degree of precision.

Subsequent to this peak, the model's performance plateaus (second plot, above line **blue line)**, thereby displaying tendencies of overfitting to the training data. Consequently, the training was reduced at 300 epochs to prevent any further overfitting.

Recall (B)

The Recall curve (B), with "B" representing the Bounding Box, quantifies the model's capacity to recognize objects deemed relevant within the images. It can be computed by taking the ratio of True Positives to the sum of True Positives and False Negatives.

This metric holds significanceas it relates to which objects classified as relevant are successfully identified within the dataset images.

An upward trajectory of this metric over the course of training is a positive sign, indicating the model's progressive improvement in the task of comprehensively detecting objects within images. On the other hand, a reversal of this trend - a downward trend in the recall curve (B) beyond a certain point - can serve as an indicator of potential overfitting of the model to the training data set.

Figure 27 correctly illustrates the Yolov8 recall curve(B). This curve represents the peak reached by the model at various times during the training process. The beginning of the curve at a lower level, followed by a progressive rise, signifies a favorable trajectory. This trend denotes the model's evolutionary progress in achieving more accurate and comprehensive detection of all the objects in the images.

The curve (second plot , above line, **orange line**) attains its peak with a recall value approaching 0.95, a good benchmark that underscores the model's aptitude in detecting objects within the images with satisfatory precision.

In totality, the Yolov8 detect recall(B) curve holds potential, exemplifying the model's gradual enhancement in accurately detecting all objects depicted within the images over the course of training iterations.

mAP50 (B)

The Mean Average Precision (mAP) serves as a widely adopted metric for evaluating accuracy in various models, including YOLO (You Only Look Once). Its assessment involves a comprehensive consideration of concepts like Intersection over Union (IoU), Precision, Recall, the Precision-Recall Curve, and Average Precision (AP), it can be seen in more detail: Gad (2020).

IoU quantifies the overlap between anticipated bounding boxes and actual ground truth bounding boxes.

mAP is derived by averaging the AP scores across diverse classes or categories. Its numeric range from 0 to 1, where higher values correspond to better model accuracy. However, it's imperative to recognize that interpreting mAP in isolation is insufficient. It should be

examined alongside other pivotal metrics, such as loss, to globally assess a model's overall performance. mAP50 computes the average precision by maintaining a constant Intersection over Union (IoU) threshold of 0.5. The utilization of an IoU threshold of 0.5 is accepted in numerous object detection trainings, as it determines whether a predicted bounding box qualifies as a true positive. mAP50 assesses the model's precision and recall at the specified IoU threshold of 0.5, thereby furnishing a comprehensive evaluation of its performance.

The mAP50 score exhibits a positive upward trend in figure 27 (third plot , above line **blue line**) as time progresses, which is indicative of favorable progress. This signifies that the model's capacity in object detection within the images is enhancing, irrespective of the objects' dimensions or positions.

mAP50-95 (B)

The distinction between this metric and the preceding one lies in the difference of the Intersection over Union (IoU) value. Unlike the fixed IoU of 0.5, this metric encompasses IoU values ranging from 0.5 to 0.95. Within this IoU range, both precision and recall are computed, subsequently culminating in the determination of the average precision. mAP50-95 offers a more comprehensive evaluation of the model's performance across a spectrum of IoU thresholds. This complex evaluation provides an enhanced comprehension of how the model's detected objects correlate with the actual ground truth boxes across varying degrees of overlap. In the provided image 28 , the concept of Intersection over Union (IoU) is visually demonstrated, accompanied by an explanation of its significance:

- **Ground Truth Bounding Box** - This is the actual box that accurately refers to the object in the image. It's typically provided in the dataset and serves as a reference for evaluating the model's accuracy. The model's predicted bounding box is compared to the ground truth bounding box to calculate metrics like IoU and precision.
- **Predicted Bounding Box** - This is the box that an object detection model predicts as the region where an object is located in an image. It's represented by coordinates that define the top-left and bottom-right corners of the box. These boxes are often displayed in different colors (like green) to differentiate them from other elements.
- **IoU** - IoU measures the degree of overlap between the two bounding boxes and serves as a measure of their spatial alignment. IoU is calculated by dividing the area of intersection between the predicted bounding box and the ground truth bounding box by the area of their union.

A higher Intersection over Union (IoU) value indicates a closer alignment between the predicted bounding box and the ground truth bounding box in object detection tasks.

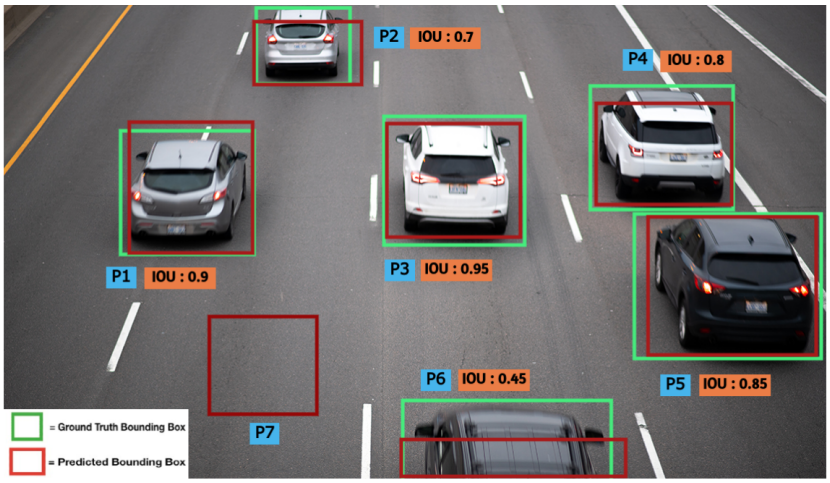


Figure 28: Example Ground Truth, Predicted boxes and IoU

The trajectory of this metric, as depicted in the graph of the figure 27 (third graph, top line, marked by the orange line), exhibits an upward trend. This upward trend persists until approximately epoch 270, at which point it plateaus.

Learning Rates (PG0 and PG1)

The learning rates (PG0 and PG1) correspond, respectively, to the learning rates of the hyperparameters lr0 and lrf, as elucidated previously. The lr0 parameter designates the initial learning rate, while lrf represents the final learning rate relative at the last epoch of training. As discernible from the graph, the learning rates exhibit a descending pattern, which is indicative of positive progress, signifying the model's learning process over the epochs. In this context, these metrics suggest that the model could potentially continue to "learn" further. However, achieving this may necessitate an increase in the number of images within the dataset. This consideration emerge from observations related to the mAP, Precision, and Recall metrics, which hint at the possibility of the model approaching overfitting, justifying the termination of training at epoch 300.

Subsequently, practical instances of inferences will be showcased using images sourced from the Tidy City database. These images have undergone analysis to confirm the model's efficacy in real-world scenarios.

Figure 29: Prediction made with the generated model, on one of the new images taken from the Tidy City application database.

As discernible from the image 29, the inference presented was **executed with a stipulated confidence threshold of 0.7**. This criterion dictates that an object is considered a waste container only when its corresponding confidence score surpasses or equals the threshold value of 0.7. A notable observation is the evident confidence scores exhibited within the red-bordered bounding boxes, all of which surpass the 0.9 threshold. This outcome signifies a satisfatory accomplishment.

Moreover, it is noteworthy that each waste container depicted in the image has been accurately identified. This consistency of accurate identifications further accentuates the satisfying nature of the model's predictive capabilities.

The model successfully inferred distinct categories of municipal waste containers, as evident from the subsequent figure 30. Notably, the predictions maintained a robust level of confidence, with all three waste containers exhibiting confidence scores exceeding the threshold of 0.9.

Figure 30: Two different predictions, one in 3 underground containers(left) and another in a litter bin (right).

6.2.2 *Results of the training with Instance Segmentation Dataset*

The subsequent sections will elucidate and expound upon the outcomes of the training procedure utilizing the dataset created for municipal waste container identification via instance segmentation. This training process aimed to preselect frames within the Tidy City database and will be comprehensively detailed and discussed.

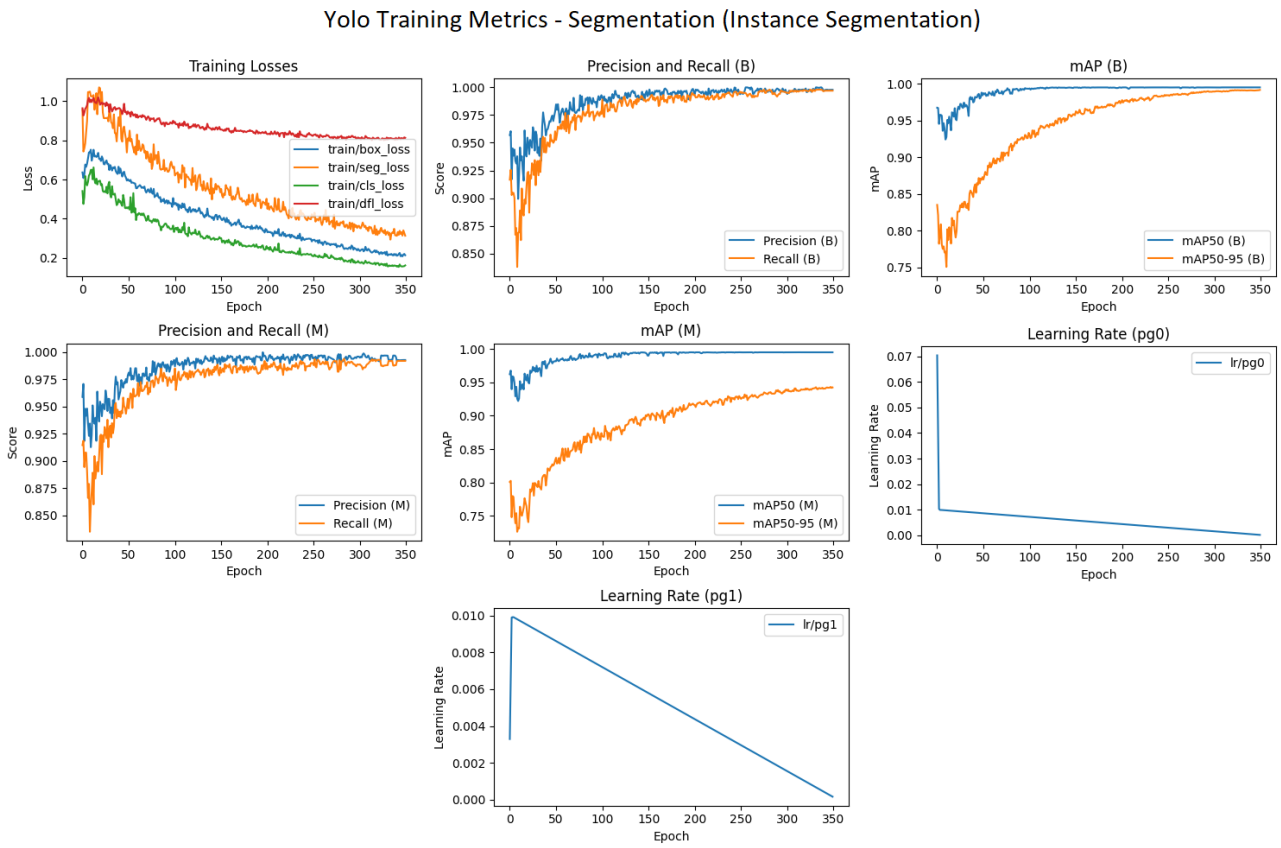


Figure 31: Best Results of the train with Instance Segmentation dataset

Box Loss

As previously indicated, this metric corresponds to the descending trajectory of the loss function across successive iterations or training epochs. The graph in figure 31 (first plot , above line **blue line**) exhibits a diminishing pattern over time, signifying the model's progressive learning process. The achieved value is considered satisfactory.

Seg Loss

The curve in figure 31,in the segmentation metric (first graph, top line, denoted by the **orange line**) serves to quantify the similarity between the predicted segmentation map and the ground truth map. This metric assesses how competent the model executes the instance segmentation task.

The trend of this metric is also characterized by a decrease over time, signifying that the model is progressively improving its performance in the instance segmentation task.

Other Loss Metrics

The reduction in the remaining two loss metrics in figure 31, cls and dfl, over the training duration is a positive indication of the model's learning and enhancement.

Nevertheless, it's worth noting that fluctuations are observable in all the loss function curves. These fluctuations may be attributed to the relatively low batch value. Regrettably, due to hardware limitations, increasing the batch value was not a feasible option.

mAP (B) and Precision and Recall (B)

These metrics hold the same significance as in the previous training, with the "(B)" indicating the performance metric of Bounding-Box achieved by the model during the training process.

Starting with the mAP metric can be seen in figure 31, both curves, represented by the model located on the far right in the top row, exhibit an upward trajectory. The values of mAP50 and mAP50-95 are notably high, ranging between 0.95 and 1. However, it's essential to acknowledge that these values are expected to decrease when applied to real-world frames. Nevertheless, they tend to form relatively straight lines, which could be suggestive of overfitting. This phenomenon is more frequently observed in small datasets, as is the case here. Consequently, the values of these curves were carefully monitored at intervals of every 50 epochs throughout the training process.

The Precision and Recall curves (second graph, top line) closely resemble the previous ones. They display an ascending pattern that tends to plateau, which suggests the possibility of overfitting.

mAP(M) and Precision and Recall(M)

These metrics hold the same significance as in the previous training, with the "(M)" refers to "mask" , which is the segmentation mask of the objects in image.

In terms of the mAP graphs, the mAP50 value closely resembles that of mAp50(B). It rises and then levels off around the values of 0.95-1. However, the mAP50-95(M) values are lower. Although the curve behaves similarly to that of mAP50-95(B), it grows and then stabilizes, with maximum values reaching 0.9-0.95.

As demonstrated in figure 32, one of the predictions generated for an image from the Tidy City dataset, with a **minimum confidence threshold of 0.7**, successfully identifies multiple containers of diverse colors, achieving satisfatory confidence scores (greater than 0.9).

Figure 32: Waste Containers detection in segmentation training.

The model has also successfully detected other types of public waste containers, as illustrated in the image 33 .

Figure 33: Litter bin detection in segmentation training.

6.2.3 *Choosing the Trained Model*

Following the completion of training for both models, a comparison was conducted using several comparable metrics. The point at which each model exhibited its best performance was analyzed, and specific metrics that provided stronger insights than others were selected for evaluation. The following order of importance is justified based on their relevance to real-world tasks:

- **mAP50-95 (Mean Average Precision over IoU Range)** - This metric is prioritized as it offers a comprehensive assessment of both detection and segmentation quality. mAP50-95 offers insights across different IoU thresholds. It assesses the model's

ability to perform consistently across varying object overlaps, adding to the robustness evaluation.

- **mAP50 (Mean Average Precision at 50 percent IoU)** - Following closely, mAP50 strikes a balance between precision and recall, being essential for a thorough evaluation of the models.
- **Precision** - Precision measures prediction accuracy, which is especially critical when minimizing false positives is a primary objective.
- **Recall** - The model's ability to capture all relevant instances, as indicated by recall, is of significant importance.
- **Box Loss** - Box loss holds a important position among the metrics due to its direct influence on the accuracy of object localization. Accurate bounding box predictions are fundamental in tasks like object detection and segmentation.
- **Class Loss** - Although important, class loss is somewhat less critical than localization accuracy. It assesses the precision of object classification, which remains vital but may not always take precedence over localization accuracy.
- **DFL Loss** - DFL loss, if applicable, is task-specific and therefore ranks the lowest in universal importance among these metrics.

Metric	Detection Model (Bounding-Box)	Segmentation Model
Epoch	128	243
mAP50	∼0.99	∼0.99
mAP50-95	∼0.88	∼0.98
Precision	1	∼0.99
Recall	∼0.99	∼0.99
Box Loss	∼0.70	∼0.29
Class Loss	∼0.41	∼0.22
DFL Loss	∼0.83	∼0.99

Table 7: Comparison of the best epoch between the two model's.

As evident from the table 7, the obtained values are high and similar to each other. It's important to note that these are training values within a controlled environment, and it's expected that these values will decrease when applied to the prediction of real-world images. An influencing factor for these high values could be the **single-class detection task in both models**. Single-class tasks are typically easier to address as they don't require distinguishing between various categories of objects. The absence of multiple classes and

complex scenarios simplifies the problem, making it more manageable for the models. In comparison, multi-class detection tasks often come with more challenges. The **segmentation model demonstrated better performance in training**, despite achieving a similar accuracy to the bounding box model. It managed to obtain lower loss values. However, to validate these results, both models were put to the test using real-world images. In the context of results obtained from image frames that were not included in the training dataset, it was observed that **when considering the outcomes on image frames that were never utilized during training, the detection model employing the bounding-box technique demonstrated superior performance**. This paradox can be elucidated by the inherent characteristics of the task requirements. Specifically, the nature of the task, whether it involves segmenting or detecting objects, can significantly impact the model's performance. Certain tasks align better with one approach over the other. In this particular instance, the objective was to discern the presence of waste containers. Object detection is well-suited for situations where the goal is to identify objects within their contextual surroundings, without necessitating an exact pixel-level mask. Another factor that contributed to the favorable outcome in segmentaton training, contrasted with the less favorable result in a real-world context, was the detection of overfitting during segmentation training. This phenomenon caused the model to become overly specialized in the training data and less adaptable to diverse, unseen scenarios. In the picture 34 , examples are presented of incorrect detections (using the segmentation technique) with a confidence score greater than or equal to 0.7.

Figure 34: Wrong Detections in using segmentation model.

In the image 34, from left to right, the model incorrectly identifies a concrete pillar as a container (confidence score of 0.77), a yellow taxi (score of 0.73), a concrete pillar of a bridge (score of 0.93), and a handrail (score of 0.77).

The model trained with the bounding-box annotation method, as mentioned earlier, performed better but also exhibited some false positive results, as can be seen in figure 35.

Figure 35: Wrong Detection using Bounding-Box annotation method.

The predictions had a high confidence score, which is understandable given the object's resemblance to a container. However, this model did not classify the previous objects(in figure 34) as public waste containers.

After a analysis of these factors, **the detection (Bounding-Box) training model was then selected.**

6.3 Results of the last training sessions

This section is dedicated to the presentation and examination of the outcomes obtained during the initial phase of the most recent training sessions. The objective is to detect various

types of containers, assess their condition, and identify the waste situated in proximity to them.

Note: For more detailed information about the training methodology, please refer to the preceding Chapter 5), specifically in the Section 5.7, and in the subsection,Section 5.7.1.

6.3.1 *Results of Initial Phase*

The conclusive results of this training endeavor are visually depicted in the following figure, figure 36:

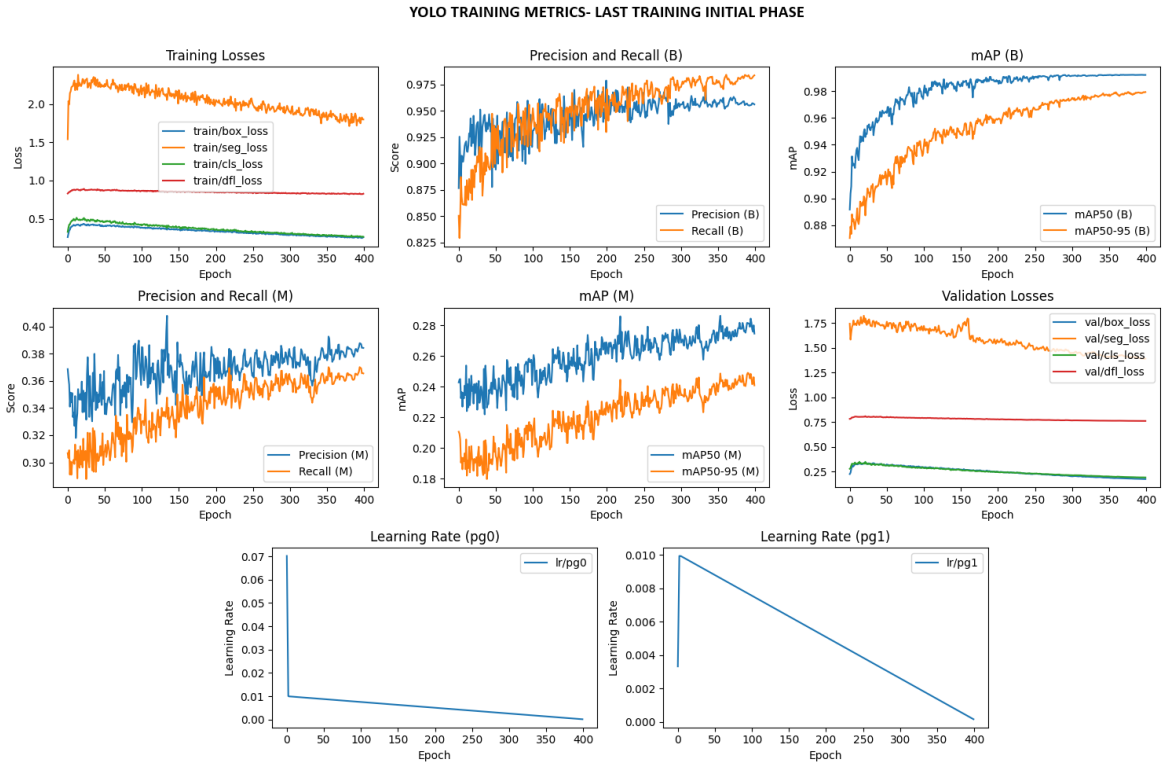


Figure 36: Results of Initial Training Phase.

Loss Trends

The provided graphs in figure 36 include two categories of losses: "Training Losses" and "Validation Losses." The "Training Losses" graph illustrates all the loss metric's values incurred over the training epochs, demonstrating a consistent decline. This decline signifies that the model is progressively learning and enhancing its predictive capabilities.

In parallel, the "Validation Losses" graph exhibits a trend similar to that of the training losses. This convergence between the validation and training losses is indicative of a positive generalization. Furthermore, it suggests that the model did not exhibit significant overfitting on the training data.

Nevertheless, even though there is an overall decreasing trend in the losses, it's noteworthy to examine the red line in both graphs in figure 36, representing the "DFL Loss." In these cases, the decline is marginal. This behavior underscores the limited variability within certain classes in the dataset, **highlighting the need to identify which classes require a greater number of samples to address this issue.**

Bounding Box Metrics

The analysis of bounding box metrics in figure 36 results can be conducted by referring to the "Precision and Recall (B)" and "mAP(B)" graphs. The obtained values consistently exhibit high precision and recall rates, **although there is still potential for improvement, suggesting that the model can continue to enhance its performance.**

This pattern is also reflected in the mAP values, both of which display an upward trajectory, signifying progress across the epochs. This trend underscores the model's capability to achieve high precision at varying IoU (Intersection over Union) thresholds.

Segmentation Metrics

The segmentation metrics are demonstrated in the graphs in figure 36 for "Precision and Recall (M)" and "mAP(M)." These metrics exhibit an upward trend across the epochs, but it **is evident that they consistently produce lower values when compared to their counterparts in the bounding box metrics.**

This observation is consistent across both graphs and **implies that the model requires additional training epochs** to enhance its capacity for accurately predicting masks on various objects within the images.

Learning Rate

Lastly, the Learning Rate graphs, present in figure 36 depict a decreasing trend over time, which aligns with the anticipated behavior when training a deep learning model. This behavior confirms that the learning rate is adjusting effectively during the training process.

In summary, it was determined that the model **performed well in bounding box detection**. However, **for segmentation tasks, it requires additional training and analysis**. To address these challenges, inferences were made using a subset of images from the Tidy City application dataset to gain insights into the difficulties the model encounters in detecting

certain classes. As an example, let's take a closed plastic 4-wheel container. In this case, the model is tasked with identifying three classes: the container type (4wheel), the waste type (4 wheel yellow), and the container's state (4 wheel yellow closed). The test results for this model demonstrated that it performs reasonably well in identifying the container type (4wheel). However, beyond that point, it encounters increased difficulty in correctly identifying the waste type (4 wheel yellow) and the container's state (4 wheel yellow closed).

It became evident that increasing the number of images in the dataset and augmenting the samples for certain classes (such as underground containers and their various types, litter bins, 2-wheel containers and their variations, as well as waste adjacent to urban waste containers, notably household appliances) is necessary to improve the model's performance in these areas.

6.3.2 *Results of Final Phase*

The final training sessions reached a plateau, and the results are depicted in the graphs below, figure 37:

Figure 37: Results of Final Training Phase.

Loss Trends

As before, there are two graphs in figure 37 that illustrate the training loss metrics: the "Training losses" graph displays the loss values of the four metrics during the training process, while the "Validation Losses" graph shows the loss values when the results are validated after each epoch.

Throughout the training process, **the training loss values seem to fluctuate, and although there is a slight decrease, it is not consistently decreasing. This suggests that the model is not converging as intended.**

Regarding the validation losses, they allow us to monitor the model's performance on an unseen dataset. Upon closer examination, it becomes apparent that, overall, **all the metrics tend to decrease**. However, as seen in the orange graph, despite this downward trend, the seg loss (corresponding to segmentation losses) remains quite high. There was no improvement (compared to the training results from the first phase) in the dfl loss curve (in red), suggesting the the continued need to analyze the classes that require more training samples.

The other two losses, box loss and cls loss (colored blue and green, respectively), showed acceptable results, with consistent values throughout training and lower loss values.

Bounding Box Metrics

The results for bounding boxes performed better compared to segmentation.

Starting with the "Precision and Recall (B)" graph in figure 37, Precision measures the accuracy of bounding box predictions, indicating how many bounding boxes are correctly marked, while Recall measures the model's ability to identify and detect objects in the image. In the graph in the figure 37, is possible to observe fluctuations in the values of these two metrics. The maximum Precision value reached during the training process was approximately 0.8, signifying that at that point, around 80 percent of the predicted bounding boxes were correct. As for Recall, the highest value achieved was 0.6, indicating that the model detects approximately 60percent of the objects.

At epoch 129, Precision decreased slightly to 0.67, while Recall increased to 0.57. This suggests a trade-off between precision and recall, where the model is becoming slightly more conservative in predicting bounding boxes but is detecting more objects.

Regarding mAP (Mean Average Precision), as shown in the "Map (B)" graph, both mAP50(B) in blue and mAP50-95 (B) in orange exhibit an increasing trend. Initially, mAP at IOU 0.5 is low, with values dropping by epoch 50. However, from epoch 100 onwards, this metric begins to recover, suggesting that the model is learning to detect objects.

By the end of training at epoch 129, mAP50(B) has improved and is closer to its initial value, with the maximum value reaching around 0.65. Similarly, mAP50-95 exhibits a

comparable behavior, with fluctuations and an increasing trend. The initial and final values are low (0.58 and 0.57, respectively), and the model reaches its maximum at around 0.6.

In summary, these two metrics fluctuate, but the overall trend indicates that the model is learning to detect objects more accurately as the training process progresses. Additional training time is required to further improve performance.

Segmentation Metrics

Analyzing the segmentation results present in the image 37, and starting with the "Precision and Recall (M)" graph, it's evident that the values fluctuate during the training process and remain generally low, with no significant upward trend. The maximum Precision obtained throughout the training process is approximately 0.4. In terms of Recall, the highest value achieved is around 0.15.

Regarding the "MAP(M)" metrics, both values, mAP50 and mAP50-95, are exceedingly low and not yet suitable for making real-world predictions. These results indicate that further training and refinement are necessary to improve the model's performance in segmentation tasks.

The obtained results indeed fall short of expectations, particularly in the context of segmenting containers within images.Additional training, an increased number of container samples, and greater variability are essential requirements for improving the model's performance. Nevertheless, it is worth noting that preliminary tests conducted on images (38, 39, 40, 41, 42) not included in the dataset have generating more promising results than initially anticipated (confidence score of 0.7 or above).

Figure 38: 4 wheel detection using the best model.

In the image 38, the model has effectively identified all the containers present. Specifically, it has correctly detected **five instances of the "4wheel" container class**, further distinguishing between the various types, **including one "4wheel blue," one "4wheel yellow," one "4wheel green," and two "4wheel normal" containers**. Moreover, the model has accurately predicted the condition of these containers, indicating that they are all in a closed state.

Regarding other types of containers, the model's performance was less satisfactory. While it can reasonably identify the primary class of container (e.g., 4-wheel, 2-wheel, etc.), its performance on other aspects, such as determining the specific waste category associated with the container, proved to be more challenging.

Figure 39: Underground Container detection using the best model.

In the image 39, the model correctly identifies two underground containers. However, the rest of the predictions are inaccurate. It incorrectly identifies normal underground containers as two blue containers, when they should be classified as underground containers for unsorted waste. Additionally, it correctly identifies the container as closed, however is not blue containers (it should be underground container normal closed).

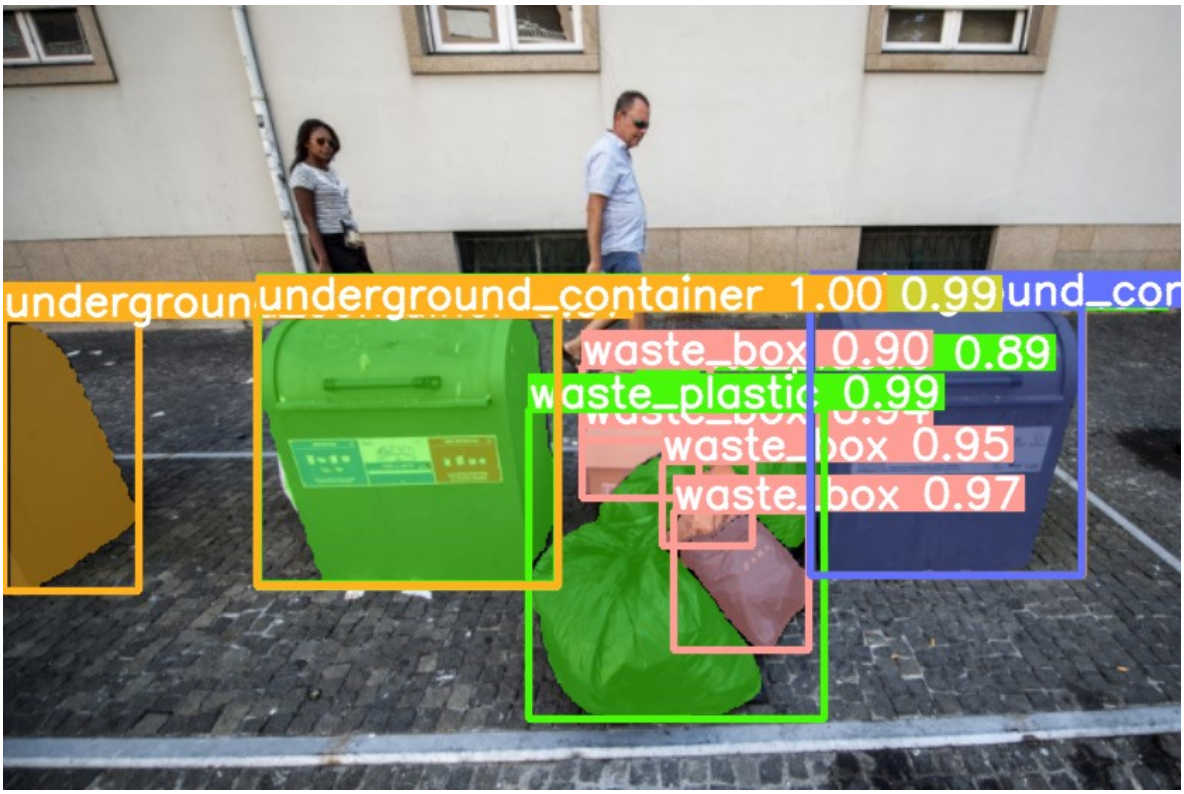


Figure 40: Underground Container with adjacent waste detection of the best model.

In another example (figure 40), with containers of the same type but arranged differently and with adjacent waste, the model performed well. It correctly identified:

- 3 containers as underground containers.
- 1 closed blue container (2 correct classes).
- 1 normal container as closed undifferentiated.
- 4 adjacent waste items as waste boxes (including 1 box and 3 paper bags).
- 2 adjacent waste items as waste plastic (2 plastic bags).

The model was also able to correctly identify litter bins, along with adjacent garbage, as shown in the image 41. However, there was a detection error in the picture, where the model incorrectly identified an object as plastic waste (with a confidence score of 0.72) when it was not.

Figure 41: Litter bin with adjacent waste detection of the best model.

The analysis also revealed the areas where the model struggled the most in detecting objects, specifically with **domestic bins and 2-wheel containers.** Regarding domestic bins, the model often failed to identify them within some images.

In the case of 2-wheel containers, the model's performance was inconsistent (as can be seen in the figure 42), sometimes successfully detecting them and other times failing to do so. Additionally, it frequently misclassified the state of these containers, often identifying them as closed when they were actually full. These findings underscore the need for additional training with more samples of these specific container types to improve model accuracy.

Figure 42: Some containers poorly (or not detected at all) detected by the model.

In summary, after a thorough analysis based on real-world examples, the model demonstrates reasonable performance and can accurately identify many types of containers. However, it exhibits varying degrees of competence in distinguishing between different container types. **Specifically, the model performs better at identifying 4-wheel containers, semi-underground containers, and litter bins compared to domestic bins, underground containers, and 2-wheel containers.** This suggests the necessity for further training and the incorporation of a larger dataset to enhance its performance across all container categories.

The model's performance in identifying waste within images is generally satisfactory, **with successful recognition of plastics, paper, household appliances, and furniture. Nonetheless, it encounters challenges in differentiating between wood and cardboard.**

Note: The analysis of these results can be seen in more detail in chapter 7, more specifically in, subsection 7.2.

7

CONCLUSION

The concluding chapter shall provide a succinct analysis of the document. The chapter will offer reflections on the work conducted, providing a conclusion. Additionally, it will encompass a discussion of lessons accumulated and expected paths for future research.

7.1 DOCUMENT REVIEW

The objectives include the development of a machine learning model designed to identify objects in images, specifically urban waste containers. The model is intended to distinguish their type (e.g., 4 wheels, 2 wheels), determine the associated waste category, and determine their condition (open, closed, or full). Concurrently, the overall aim is to promote increased recycling rates among citizens residing in the municipalities. An analysis of the context and background is made, where a historical framework is given and some important machine learning algorithms are presented to review in the development of this master's project.The following chapter, "State of Art," provides an overview of existing solutions in the market, annotation tools, types of annotation, and more. Subsequently, a proposed approach is presented through a workflow diagram to provide a cohesive plan for the execution of this project. Chapters 5 and 6 are dedicated to the development of the project and the presentation of its results, offering a comprehensive account of the entire process and detailed documentation of the outcomes.

The document concludes with a final reflection in Chapter 7.

7.2 RESULTS ACHIEVED

The primary objective of this Master's project was the detection and classification of urban solid waste containers, and this goal was successfully accomplished. A final YOLOv8 model was trained to detect and classify different containers, although varying levels of difficulty were encountered in the process.

Subsequently, the project aimed to further classify these containers based on the type of waste they contain, which was also achieved. However, some classes still present challenges in terms of accurate identification.

Additionally, the model was designed to identify adjacent waste and its composition, with a focus on common types like plastic and paper.

An important observation from the outcomes of this project is the discrepancy in the filling frequency of different containers. **For instance, 4-wheel plastic containers are filled more frequently than 4-wheel glass containers**. This disparity in filling frequency has implications for dataset creation, making it more challenging to find images of containers with specific characteristics. For example, **obtaining images of overfilled glass containers is more difficult than obtaining images of overfilled plastic containers**.

In the following tables, three classifications are used, Detects Often, Satisfactory and Needs Improvement, to demonstrate the need to improve the algorithm and/or the data set in future work.

7.2.1 *Waste Containers Class*

The table 8 shows the analysis of the model's final results for detecting container classes.

Waste Containers Class	**Analysis**
2 Wheel Container	Needs Improvement
4 Wheel Container	Detects Often
Domestic Container	Needs Improvement
Litter Container	Satisfactory
Semi Underground Container	Detects Often
Underground Container	Satisfactory

Table 8: Analysis of the final results in terms of detecting the class of municipal solid waste containers.

7.2.2 *Waste Containers Color*

The table 9 shows the results obtained for the waste contained in each class of container.

Waste Containers Class	Analysis
2 Wheel Green Container	Needs Improvement
2 Wheel Yellow Container	Satisfactory
2 Wheel Blue Container	Needs Improvement
2 Wheel Normal Container	Satisfactory
4 Wheel Green Container	Needs Improvement
4 Wheel Yellow Container	Detects Often
4 Wheel Blue Container	Satisfactory
4 Wheel Normal Container	Satisfactory
Underground Container Green	Satisfactory
Undeground Container Yellow	Satisfactory
Underground Container Blue	Needs Improvement
Underground Container Normal	Satisfactory
Semi Underground Container Green	Satisfactory
Semi Underground Container Yellow	Detects Often
Semi Underground Container Blue	Detects Often
Semi Underground Container Normal	Satisfactory

Table 9: Analysis of the final results in terms of the type of waste that a certain class of container holds.

7.2.3 *Waste Container Lid Status*

In the table 10, it's possible to see how the model behaves when it comes to detecting the state of the lid in the different container classes.

Waste Containers Class	Analysis
2 Wheel Green Container Closed	Satisfactory
2 Wheel Green Container Open	Needs Improvement
2 Wheel Green Container Overfill	Needs Improvement
2 Wheel Yellow Container Closed	Satisfactory
2 Wheel Yellow Container Open	Satisfactory
2 Wheel Yellow Container Overfill	Satisfactory
2 Wheel Blue Container Closed	Satisfactory
2 Wheel Blue Container Open	Needs Improvement
2 Wheel Blue Container Overfill	Satisfactory
2 Wheel Normal Container Closed	Satisfactory
2 Wheel Normal Container Open	Satisfactory
2 Wheel Normal Container Overfill	Satisfactory
4 Wheel Green Container Closed	Satisfactory
4 Wheel Green Container Open	Satisfactory
4 Wheel Green Container Overfill	Satisfactory
4 Wheel Yellow Container Closed	Detects Often
4 Wheel Yellow Container Open	Detects Often
4 Wheel Yellow Container Overfill	Satisfactory
4 Wheel Blue Container Closed	Detects Often
4 Wheel Blue Container Open	Detects Often
4 Wheel Blue Container Overfill	Satisfactory
4 Wheel Normal Container Closed	Detects Often
4 Wheel Normal Container Open	Detects Often
4 Wheel Normal Container Overfill	Satisfactory
Underground Container Green Closed	Satisfactory
Underground Container Green Open	Satisfactory
Underground Container Green Overfill	Needs Improvement
Undeground Container Yellow Closed	Detects Often
Undeground Container Yellow Open	Detects Often
Undeground Container Yellow Overfill	Satisfactory
Underground Container Blue Closed	Satisfactory
Underground Container Blue Open	Satisfactory
Underground Container Blue Overfill	Satisfactory
Underground Container Normal Closed	Detects Often
Underground Container Normal Open	Detects Often
Underground Container Normal Overfill	Satisfactory
Semi Underground Container Green Overfill	Needs Improvement
Semi Underground Container Yellow Overfill	Detects Often
Semi Underground Container Blue Overfill	Detects Often
Semi Underground Container Normal Overfill	Satisfactory
Domestic Bin Overfill	Needs Improvement
Litter Bin Overfill	Detects Often

Table 10: Analysis of the state of the lid of the different types of container class.

7.2.4 *Waste Classification Results*

With regard to the waste materials adjacent to the containers, the analysis of the results can be seen in table 11.

Waste Class	Analysis
Waste Box	Satisfactory
Waste Furn	Satisfactory
Waste Glass	Needs Improvement
Waste Metal	Needs Improvement
Waste Wood	Needs Improvement
Waste Plastic	Detects Often
Waste Appliance	Satisfactory

Table 11: Analysis of the detection of the different constituent materials of the waste adjacent to the municipal solid waste containers.

In summary, this Master's project has successfully achieved all its predefined objectives with some limitations. Nevertheless, as detailed in the Future Work section, there is room for improvement in the model's accuracy and classification capabilities. Achieving this improvement would necessitate more extensive training and the expansion of the dataset with additional container samples.

7.3 LESSONS LEARNED

One of the most valuable lessons learned during the development of this Master's project refer to the complete process of creating the final object detection model. Each phase, from image acquisition and dataset assembly to annotation techniques and their applicability across different scenarios, was crucial to the project's success. Familiarity with existing image object detection algorithms, the opportunity to experiment with various configurations, hyperparameter adjustments, and the utilization of libraries like CUDA were all pivotal aspects of this learning process, often involving trial and error. These experiences have provided valuable insights applicable beyond the academic realm.

Furthermore, the project gave me light on the complexity of urban waste management, underscoring its vastness and the multiple challenges it poses. Despite the complexities, numerous companies are actively engaged in addressing issues within the field, particularly in areas such as predicting container fill levels and enhancing recycling rates.

Ultimately, this endeavor has equipped me with significant expertise in a field that is rapidly gaining prominence, with implications for resolving various everyday challenges.

7.4 FUTURE WORK

In the short term, enhancements are imperative, particularly in terms of refining both the dataset and the model. The exploration of alternative object detection models may also be worth considering.

Looking ahead to the medium and long term, the project could delve into real-time detection capabilities. Moreover, extracting metadata from images (e.g., street names, timestamps) could facilitate the development of a more robust model capable of optimizing waste collection routes for municipal waste management companies. This could require predicting when specific waste containers are likely to reach full capacity, considering regional variations such as those observed during holidays.

It would also be interesting to add real-time geo-referencing functionalities, so that people can see where the containers are, and it could help to predict how full a container is if waste is shown in the image.QR Codes can also be used to identify these containers, so we can create a unique id for each container.

Furthermore, the integration of these models with sensor technology designed to measure container fill levels could aid data collection efforts and further enhance the project's impact.

BIBLIOGRAPHY

GitHub - Kinvert/CUDA-Stuff: CUDA Stuff Here — github.com. https://github.com/Kinvert/CUDA-Stuff, 2022.

United States Environmental Protection Agency. Pollution prevention and waste management, 2022. URL https://cutt.ly/7VYG8pS.

arditi. Projeto tidy city. URL https://www.arditi.pt/pt/projetos-em-execucao/projeto-tidycity.html.

PaddlePaddle Authors. Paddledetection, object detection and instance segmentation toolkit based on paddlepaddle. https://github.com/PaddlePaddle/PaddleDetection, 2019.

C40 Knowledge Hub C40 Cities Climate Leadership Group. How cities can boost recycling rates, 2019. URL https://cutt.ly/0VYWVoO.

candam. URL https://candam.eu/.

containerdetectortrain. Tugawaste dataset. https://universe.roboflow.com/containerdetectortrain/tugawaste, may 2023. URL https://universe.roboflow.com/containerdetectortrain/tugawaste. visited on 2023-05-11.

CVAT.ai Corporation. Computer Vision Annotation Tool (CVAT), September 2022. URL https://github.com/opencv/cvat.

Raquel Greice de Souza Marotta Alfaia, Alyne Moraes Costa, and Juacyara Carbonelli Campos. Municipal solid waste in brazil: A review. *Waste Management & Research*, 35 (12):1195–1209, 2017. doi: 10.1177/0734242X17735375. URL https://doi.org/10.1177/0734242X17735375. PMID: 29090660.

Ahmed Fawzy Gad. Evaluating object detection models using mean average precision (map), 2020. URL https://blog.paperspace.com/mean-average-precision/.

GeeksforGeeks. Convolutional neural network (cnn) in machine learning, 2023. URL https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/.

Jonathan Hui. map (mean average precision) for object detection, 2018. URL https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173.

IBM. What is the k-nearest neighbors algorithm? URL https://www.ibm.com/topics/knn.

Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023. URL https://github.com/ultralytics/ultralytics.

Grace Karimi. Introduction to yolo algorithm for object detection, 2021. URL https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/.

Fermin Koop. Home environment why is recycling so important? the dirty truth behind our trash, 2022. URL https://cutt.ly/4VYHenS.

Mukilan Krishnakumar. A gentle introduction to yolov8, 2023. URL https://api.wandb.ai/links/mukilan/a7a0grd4.

Garrick Louis. A historical context of municipal solid waste management in the united states. *Waste management research : the journal of the International Solid Wastes and Public Cleansing Association, ISWA*, 22:306–22, 09 2004. doi: 10.1177/0734242X04045425.

Trevor Lynn. Launch: Label data with segment anything in roboflow. URL https://blog.roboflow.com/label-data-segment-anything-model-sam/.

MathWorks. Multilabel image classification using deep learning. URL https://www.mathworks.com/help/deeplearning/ug/multilabel-image-classification-using-deep-learning.html.

David Meyer. Support vector machines. *R News*, 1:23–26, 2001. ISSN 1609-3631. https://journal.r-project.org/articles/RN-2001-025/.

Joseph Nelson. The difference between missing and null annotations, 2020. URL https://blog.roboflow.com/missing-and-null-image-annotations/.

FRED OH. What is cuda?, 2012. URL https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/.

Ashish Patel. What is object detection?, 2020. URL https://medium.com/ml-research-lab/what-is-object-detection-51f9d872ece7.

Pysource. Cpu vs gpu (training yolo v4). how much faster is the gpu? URL https://www.youtube.com/watch?v=-R17DTPLfm4.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

Facebook Research. Detectron2 model zoo and baselines. URL https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md#coco-instance-segmentation-baselines-with-mask-r-cnn.

Roboflow. Generate augmented images. URL https://docs.roboflow.com/datasets/image-augmentation.

Lior Rokach and Oded Maimon. Decision trees. *Data mining and knowledge discovery handbook*, pages 165–192, 2005.

Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way, 2018. URL https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

Husna Sayedi. Introduction to image annotation for ml and ai, 2021. URL https://www.taus.net/resources/blog/introduction-to-image-annotation-for-ml-and-ai.

sensoneo. Waste monitoring of semi-underground and underground bins. URL https://sensoneo.com/use-case/monitoring-underground-semiunderground-bins/.

Sai Shashank. Detectron2 vs. yolov5 (which one suits your use case better?). https://medium.com/ireadrx/detectron2-vs-yolov5-which-one-suits-your-use-case-better-d959a3d4bdf, 2021.

Andréia Azevedo Soares. Portugal reciclou 16,1 em 2020 e está ainda mais longe da meta europeia, 2022a. URL https://cutt.ly/5VYbDGe.

Andréia Azevedo Soares. Taxa de reciclagem em portugal mantém-se "vergonhosamente" nos 21 URL https://www.publico.pt/2022/11/11/azul/noticia/taxa-reciclagem-portugal-mantemse-vergonhosamente-21-zero-2027322.

João Sousa, Ana Rebelo, and Jaime S. Cardoso. Automation of waste sorting with deep learning. In *2019 XV Workshop de Visão Computacional (WVC)*, pages 43–48, 2019. doi: 10.1109/WVC.2019.8876924.

stateofwastebins. Type of waste containers dataset. https://universe.roboflow.com/stateofwastebins/type_of_waste_containers, aug 2023. URL https://universe.roboflow.com/stateofwastebins/type_of_waste_containers. visited on 2023-09-19.

StereoLabs. Performance benchmark of yolo v5, v7 and v8. URL https://www.stereolabs.com/blog/performance-of-yolo-v5-v7-and-v8/.

Tim Storm and Christoph Mertz. Detecting and classifying bus stop trash cans using camera-equipped public transit vehicles.

TARASFOUNDATION. A short history of solid waste management, 2020. URL https://taras.org/2020/10/10/a-short-history-of-solid-waste-management/.

Maxim Tkachenko, Mikhail Malyuk, Andrey Holmanyuk, and Nikolai Liubimov. Label Studio: Data labeling software, 2020-2022. URL https://github.com/heartexlabs/label-studio. Open source software available from https://github.com/heartexlabs/label-studio.

Ultralytics. Train arguments, a. URL https://docs.ultralytics.com/modes/train/#arguments.

Ultralytics. Object detection datasets overview, b. URL https://docs.ultralytics.com/datasets/detect/.

Ultralytics. Instance segmentation datasets overview, c. URL https://docs.ultralytics.com/datasets/segment/.

Wikipedia. Flops. URL https://en.wikipedia.org/wiki/FLOPS.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

Wanjun Xia, Yanping Jiang, Xiaohong Chen, and Rui Zhao. Application of machine learning algorithms in municipal solid waste management: A mini review. *Waste Management & Research*, 40(6):609–624, 2022. doi: 10.1177/0734242X211033716. URL https://doi.org/10.1177/0734242X211033716. PMID: 34269157.

Tony Yiu. Understanding random forest, 2019. URL https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

Jure Zupan. Introduction to artificial neural network (ann) methods: What they are and how to use them. *Acta Chimica Slovenica*, 41, 01 1994.