# Coastline Litter Detection using Deep Convolutional Neural Networks

by

Ioanna A. Rodopoulou

Supervisor: Professor Partsinevelos Panagiotis

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science in Engineering

Technical University of Crete

Chania, June 2022

# Abstract

One of the environmental problems of modern life is the ecological and aesthetic degradation of coastal zones by litter disposal and accumulation. Research conducted to monitor and evaluate pollution incidents in the coastal environment usually consists of groups of volunteers or civil servants who record, count and sort the litter. However, recent research efforts show that litter detection can become an automated process, thanks to the development of remote sensing and computer vision methods. Recording of litter by simple devices like smartphones or more sophisticated devices like drones, can be used as an input to investigate the performance of state-of-the-art object detection algorithms. In the current study, the Mask R-CNN algorithm was used to investigate its performance in detecting coastline litter and classifying it based on its material and type. Mask R-CNN is part of the R-CNN family of deep learning object detection algorithms that are based on convolutional neural networks (CNNs), and it is able to tackle two tasks of computer vision: object detection and instance segmentation. The performance of Mask R-CNN was mainly evaluated on a domain-specific image dataset created to facilitate this study. It was also tested on an open-sourced drone dataset of litter images, since no study has investigated the use of this algorithm on this specific image dataset. The experimental results in terms of average precision showed that Mask R-CNN exhibited strong potential in litter detection and segmentation of the new dataset, but performed moderately on the drone dataset because of the small size of litter. The algorithm showcased great predictions for well-represented classes, but performed poorly on others that were either under-represented or contained objects that varied significantly in shape, indicating challenges that will need to be furthered addressed.

# Περίληψη

Ένα από τα περιβαλλοντικά προβλήματα της σύγχρονης ζωής αποτελεί η οικολογική και η αισθητική υποβάθμιση των παράκτιων ζωνών από την απόθεση και στη συσσώρευση απορριμάτων. Έρευνες που διεξάγονται για την παρακολούθηση και αξιολόγηση των περιστατικών ρύπανσης στο παράκτιο περιβάλλον συνήθως αποτελούνται από ομάδες εθελοντών ή κρατικών υπαλλήλων που καταμετρούν και ταξινομούν τα απορρίματα. Πρόσφατα όμως γίνονται προσπάθειες για την αυτοματοποίηση αυτών των διαδικασίων με μεθόδους τηλεπισκόπησης και μηχανικής όρασης. Στόχος της παρούσας έρευνας είναι η αυτοματοποίηση/διερεύνηση της ανίχνευσης των απορριμάτων μέσα από εικόνες με τη χρήση βαθέων συνελικτικών νευρωνικών δικτύων (deep convolutional neural networks ή deep CNNs). Για το σκοπό της εργασίας, έγινε λήψη ενός συνόλου δεδομένων εικόνων από ακτογραμμή της περιοχής των Χανίων. Εξετάστηκε επίσης η ανίχνευση των απορριμάτων μέσα από εναέριες εικόνες. Χρησιμοποιήθηκε ένας από τους πιο δημοφιλείς στην επιστημονική κοινότητα αλγορίθμους ανίχνευσης αντικειμένων, ο Mask R-CNN, ο οποίος εκπαιδεύτηκε για την ανίχνευση και οριοθέτηση κάθε διακριτού απόρριματος που εμφανίζεται σε μια εικόνα. Η λειτουργία του αλγορίθμου βασίζεται σε τεχνικές βαθιάς μάθησης που αποτελούνται από τη σύνθεση τεχνητών νευρωνικών δικτύων τα οποία εκπαιδεύονται για να προβλέπουν με όσο το δυνατόν περισσότερη ακρίβεια την τοποθεσία και την αναγνώριση του αντικειμένου στην εικόνα. Συγκεκριμένα, ο αλγόριθμος αξιοποιεί βαθιά συνελικτικά νευρωνικά δίκτυα τα οποία δέχονται κατάλληλα επεξεργασμένες εικόνες με απορρίματα από το παράκτιο περιβάλλον και τις αναλύουν με σκοπό την εύρεση των μοτίβων που θα τα βοηθήσουν να κάνουν σωστές προβλέψεις αναγνώρισης και οριοθέτησης του αντικειμένου.

Ο Mask R-CNN αλγόριθμος αποτελείται από τρία κύρια τεχνητά νευρωνικά δίκτυα. Το πρώτο δίκτυο υποδέχεται τις εικόνες και είναι υπεύθυνο για την εξαγωγή των χαρακτηριστικών τους. Αυτό το νευρωνικό δίκτυο αποτελείται από ένα προ-εκπαιδευμένο νευρωνικό δίκτυο (ResNet101) από το οποίο έχουν χρησιμοποιηθεί μόνο τα μέρη του δικτύου που εξάγουν χαρακτηριστικά ενώ τα υπόλοιπα έχουν αποκοπεί. Το νευρωνικό δίκτυο παράγει εικόνες-χάρτες αποτύπωσης χαρακτηριστικών της κάθε εικόνας που εισέρχεται σε αυτό σε πολλαπλά επίπεδα κλίμακας δομής πυραμίδας. Το δεύτερο νευρωνικό δίκτυο σαρώνει τους χάρτες χαρακτηριστικών σε όλα τα επίπεδα κλίμακας και προβλέπει ποιες περιοχές περιέχουν απορρίματα. Το τρίτο νευρωνικό δίκτυο χωρίζεται σε δύο παράλληλα μέρη. Το ένα μέρος χρησιμοποιεί τις περιοχές που εντόπισε το προηγούμενο νευρωνικό δίκτυο για να προβλέψει για κάθε απόρριμα το πλαίσιο οριοθέτησής του και την τάξη στην οποία ανήκει. Το άλλο μέρος του νευρωνικού δικτύου παράγει μια δυαδική μάσκα που οριοθετεί το σχήμα του κάθε απορρίματος.

Η εκμάθηση του αλγορίθμου αντιμετωπίζεται ως πρόβλημα μαθηματικής βελτιστοποίησης. Προσαρμόζει τις παραμέτρους του με σκοπό την ελαχιστοποίηση της συνάρτησης κόστους για το κάθε ζητούμενο που καλείται να προβλέψει, δηλαδή την εύρεση των περιοχών της εικόνας που ενδέχεται να περιέχει απόρριμα, καθώς και την ταξινόμηση, τις συντεταγμένες του πλαισίου οριοθέτησης και τη δυαδική μάσκα επικάλυψης των απορριμάτων.

Για την παρούσα εργασία διεξήχθησαν τρία πειράματα. Το πρώτο πείραμα αφορούσε την εκπαίδευση του αλγορίθμου για την ανίχνευση απορριμάτων στις εικόνες που λήφθησαν από ακτογραμμή της περιοχής των Χανίων, και στις εναέριες εικόνες ενός συνόλου δεδομένων ανοιχτής πηγής. Τα πειραματικά αποτελέσματα έδειξαν ότι ο Mask

R-CNN επέδειξε ισχυρή δυναμική στην ανίχνευση απορριμμάτων στις εικόνες των ακτογραμμών με μέση ακρίβεια που φτάνει μέχρι και 92%, αλλά είχε μέτρια απόδοση στο σύνολο εναέριων εικόνων με μέση ακρίβεια που φτάνει το 53%. Καθοριστικό παράγοντα στη μειωμένη απόδοση ανίχνευσης στις εναέριες εικόνες έπαιξε το μικρό μέγεθος των απορριμμάτων.

Στο δεύτερο και στο τρίτο πείραμα χρησιμοποιήθηκαν μόνο οι εικόνες που λήφθησαν από την ακτογραμμή των Χανίων. Το δεύτερο πείραμα στόχευε στην εκπαίδευση του αλγορίθμου για την ανίχνευση απορριμάτων και την ταξινόμησή τους σε πλαστικά και μη-πλαστικά. Τα αποτελέσματα έδειξαν καλύτερες τιμές μέσης ακρίβειας για τα μη-πλαστικά, γύρω στο 83%, ενώ για τα πλαστικά η μέση ακρίβεια έφτασε το 72%.

Στο τρίτο πείραμα ο στόχος ήταν η εκμάθηση του αλγορίθμου να ανιχνεύει απορρίματα και να τα ταξινομεί σε μπουκάλια, ποτήρια μιας χρήσης, μεταλλικά δοχεία και άλλα. Ο Mask R-CNN ανίχνευσε τα μπουκάλια και τα μεταλλικά δοχεία με πολύ καλή μέση ακρίβεια με μέγιστες τιμές να φτάνουν 88%. Με τις υπόλοιπες κατηγορίες απορριμάτων παρουσίασε πρόβλημα, καθώς έδειξε να μπερδεύει τα ποτήρια μιας χρήσης με τα μεταλλικά δοχεία και τα υπόλοιπα απορρίματα με αντικείμενα του υποβάθρου των εικόνων. Το σχήμα των απορριμάτων και ο αριθμός των δειγμάτων σε κάθε κατηγορία ταξινόμησης αποτέλεσαν καθοριστικούς παράγοντες στην απόδοση του αλγορίθμου, υποδεικνύοντας προκλήσεις που θα πρέπει να αντιμετωπιστούν σε μελλοντικές εργασίες.

# List of Figures

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

Each year the marine and coastal environment is gradually impacted due to accumulated waste. Coastal pollution is associated with several ecological, social, and economic problems to the extent that is endangering human health, wildlife, and reducing the availability of ecosystem goods and services. Some of the various problems caused by this pollution include the adverse effects on the animals digesting the litter, the movement of litter from coastal areas into the water, and the hazardous effects on the marine ecosystem by non-biodegradable litter (Asensio-Montesinos et al., 2019). The economy and human health are affected in terms of sustainable and safe fisheries and aquaculture, recreation and heritage values. Significant expenses associated with, current and future, potential ecosystem degradation are expected. Beach areas, which are critical components of the coastal systems from an environmental and economic point of view, are affected by illegal litter dumping, resulting in utility value reduction (Barboza et al., 2019). Littered environments repel visitors and directly affect local businesses which spend time and financial resources to clean.

Litter composition in the beach areas is a complex phenomenon and the severity of the problem depends on both human actions and the environment. Surveys of marine and coastal litter frequently identify plastic as the major component, contributing from 60% to 80% of the total with varying polymer chemical compositions (Barboza et al., 2019; Erni-Cassola et al., 2019). Despite initiatives to monitor and reduce plastic litter, such as from the United Nations Environmental Programme (UNEP) and the European Marine Strategy Framework Directive, there is still a long way to manage it effectively (Asensio-Montesinos et al., 2019; Galgani et al., 2019).

In the coastal environment, the prominent sources of litter are human recreational activities, smoking, waste dumping into the beach or the sea (Pervez et al., 2020). The fact that more and more litter concentrations are found contaminating natural environments made monitoring a common undertaking for academic, government, and environmental organizations. Since most marine litter originates from land-based sources, surveying the litter from beach areas can be used as a tool for monitoring litter pollution (Asensio-Montesinos et al., 2019).

Shoreline surveys are generally considered to be the simplest, inexpensive, and the most direct way to monitor stranded litter compared to surveys in other environmental compartments. The process of detection, counting and classification of litter is carried out manually usually by volunteers or civil servants. But with the development of the object detection technology, this process can be automated to support research and citizen science.

## 1.2 Current methods

Litter items are collected by environmental-related entities for analysis that includes counting, classification and categorization of materials. There are sampling programs designed to monitor standardized macro-litter ($> 25mm$), especially plastic, like the OSPAR and MARLIN monitoring programs, commonly used in the North-East Atlantic coastlines and the Baltic region (OSPAR, 2010; MARLIN, 2013). These programs require training of groups of people to carry out the process of detection-counting-classification of litter items. But with the abrupt development of object detection technology, this process can be performed in a more automated manner. The tedious work of people monitoring garbage in the environment can be accomplished with remote sensing systems, and object detection algorithms can take on the task of detecting, counting and classifying trash items.

In the last couple of years, researchers are working towards developing an efficient and scalable environmental monitoring system. Depending on the target environment, they are tracking litter items by installing networks of cameras (Olivelli and Rosebrock, 2020), using mobile cameras (Ping et al., 2020), aerial (Kraft et al., 2021) or underwater visual systems (Fulton et al., 2019). They train object detection models to find and classify trash in the imagery or video acquired by the surveillance system in the field. There are several object detection models that are trained and evaluated using benchmark image datasets like the famous ImageNet Challenge (Russakovsky et al., 2015), PASCAL VOC Challenge (Everingham et al., 2015), Microsoft's COCO Challenge (Lin et al., 2015), and Open Images Challenge (Google Research, 2019). Research is ongoing towards optimizing object detection algorithms aiming at better detection speed and accuracy. Current top performing object detectors employ detection proposals to guide the search for objects (Girshick et al., 2014; Girshick, 2015; Ren et al., 2017; He et al., 2017). This kind of algorithms will be discussed extensively in the third chapter.

## 1.3 Object detection

Object detection technology is selected as the key discipline in this study because it has made remarkable progress in many directions due to the rapid advances of deep learning algorithms and the computing power of GPUs in the last decade (Chollet, 2017). Object detection is known as the computing capability in computer vision which aims to locate, recognize and differentiate targeted image objects. It involves training computers to understand and interpret imagery. It is underpinned by a subset of AI called deep learning that seeks to imitate how the human brain processes data based on artificial neural networks.

Today, object detection in images is widely adopted and enhanced using algorithms with many specialized layers for automating the feature extraction process. These learning algorithms, characterized as "deep" because of their depth in layers, are able to learn to extract features from the images by using convolutional neural networks (CNNs) in their architecture. CNNs act as a specialized filtering process in an image, where repeated filtering on an input image results in a map of activations, called a feature map, indicating the locations and strength of a detected feature in the image. The neural network has the ability to adjust itself to reduce the error in detection, and thus, improve or learn on each own how to extract the information for the task at hand (Chollet, 2017).

The current thesis aims to provide further insights when it comes to trash detection with CNN-based algorithms. Approaches like the object detection algorithms from the Region-CNN (R-CNN) algorithm family, discussed extensively in the third chapter, are in favor for this project because they have evolved to integrate the object detection process into a single neural network, simplifying training and inference, but also increasing the processing speed significantly (Ren et al., 2017; He et al., 2017).

## 1.4 Objectives

The present research examines alternative methods of litter detection in coastal areas and intends to provide valuable technological insights for litter pollution monitoring studies and cleanup operations. The present research examines alternative methods of litter detection in coastal areas and intends to provide valuable technological insights for litter pollution research studies and cleanup operations. Automated litter detection in images collected from the region of interest can help assess litter occurrences in the environment. Image acquisition can be done either through in situ ground assessments or through remote sensing methods. Prior studies have extensively used unmanned aerial vehicles for shoreline litter assessments (Fallati et al., 2019; Gonçalves et al., 2020; Lo et al., 2020; Merlino et al., 2020). The reasons are centered around the advantages provided by aerial remote sensing, such as the high spatial resolution and the high area coverage, that facilitates the monitoring of litter. Therefore, part of this research focuses on processing aerial images containing litter distributed in urban areas. But the biggest part of this work focuses on processing litter imagery obtained via mobile device during ground assessment of coastal areas. The reason was to compare the performance of a deep learning algorithm for locating and classifying litter in images collected from the shoreline assessment with the performance of the same algorithm applied on aerial images.

In the context of litter detection, this study aims to provide the scientific community with an evaluation of the effectiveness of a state-of-the-art deep learning object detection algorithm for litter detection on coastal regions. Based on these, this research breaks down to achieve the following objectives:

1. Evaluation of the performance of one of the most prominent algorithms for object detection and instance segmentation in images, namely Mask R-CNN (He et al., 2017).

2. Comparison of the efficiency of the algorithm in litter detection in different datasets.

3. Comparison of the efficiency of the algorithm in locating and categorizing litter by material and type.

The next chapter states the advances and limitations in recent technologies and monitoring strategies for litter detection. Chapter 3 presents the theoretical background of this study emphasizing the understanding of the basic concepts related to the operation of the algorithm used.

Chapter 4 outlines the experimental work carried out and Chapter 5 discusses in detail the results. The final chapter states the conclusions of this study while summing up the main results.

# Chapter 2

# Literature Review

## 2.1 Introduction

In recent years, a variety of research efforts have been made to track waste in the environment using the computer vision task of object detection in images due to remarkable advances in machine learning methods. These recent efforts are usually accompanied by remote sensing methods to collect the necessary data for object detection. In the first part of this chapter, a brief reference is made to satellite remote sensing, and a broader reference is made to aerial remote sensing which is commonly used because of the advantages that exhibits. The second part of this chapter is focused on data processing methods, applied by several researchers, for the object detection task of finding trash within images.

## 2.2 Tracking litter via remote sensing

Coastal environments can be very polluted because of the human factor. No matter the effort exerted by municipalities, waste is always present in the environment people has access to. Also, there are several cases of waste accumulating in remote areas because of physical transferring through wind, storm or wave currents. Simple solutions involve sending individuals or groups of volunteers to conduct on-ground visual counts. Unfortunately, littered locations are not always known. For example, in coastal environments, individuals can clean areas like beaches with high human traffic, while trash piles are generated in other coastal compartments. Some remote coastal areas might have accessibility and safety issues by nature like slippery rocks, precipitous slopes, etc. In these cases, remote sensing should take the lead on monitoring garbage. Remote sensing systems can make a substantial contribution to planning preparedness and mitigating potential issues of an on-ground assessment. They can provide first-hand information on changes in garbage abundance, and on abnormal amounts or "hot spots" associated with marine debris spills on the shores. Remote sensing via satellites and unmanned aerial vehicles for tracking litter items are discussed in the following sections.

### 2.2.1 Satellite remote sensing

At present, there are several remote sensing satellites providing imagery for research and operational applications. They are able to provide large area coverage, frequent and repetitive coverage of an area of interest, quantitative measurement of ground features using radio-metrically calibrated sensors, semi automated computerized processing and analysis, and relatively lower cost per unit area of coverage (Chuvieco, 2016). Hence, they potentially being ideal tools for global garbage tracking. However, there are some observational requirements needed to take into consideration when planning to use satellite remote sensing, such as identification of physico-chemical properties and their relation to a detectable signal from space. Especially for plastic litter, according to Hartmann et al. (2019), in order to classify it, it has to contain synthetic or heavily modified natural polymers as essential ingredients. The size, shape, structure, color, and origin of the object are considered secondary characteristics and not essential qualifying properties. Therefore, plastics' polymeric nature can be defined as the main observable property for a remote sensing system, The observable property should be based on the modification of the electromagnetic radiation spectrum signature due to the chemical signature of polymers (Martínez-Vicente et al., 2019). By targeting a particular polymeric compound, implies the expectation of separating the signature of plastics from all other kind of litter, either man-made or natural.

During the design of a satellite remote sensing system, the temporal and spatial resolution also needs to be determined. These sampling requirements are usually expressed as the minimum values or the threshold required for the success of the system, as well as the goal requirements which would be useful to advance the state of current knowledge. Successive iterations are expected before the selection of these values to ultimately become the engineering specifications of a satellite remote sensing system. Once these values are defined, they need to be compared with current capabilities, to signal potential suitability and knowledge gaps (Martínez-Vicente et al., 2019). Copernicus Sentinel fleet and VIIRS and Landsat series are already mature observing systems, covering various spatial scales and application domains, and it is essential to investigate its monitoring potential for trash and especially plastic pollution before looking into new solutions.

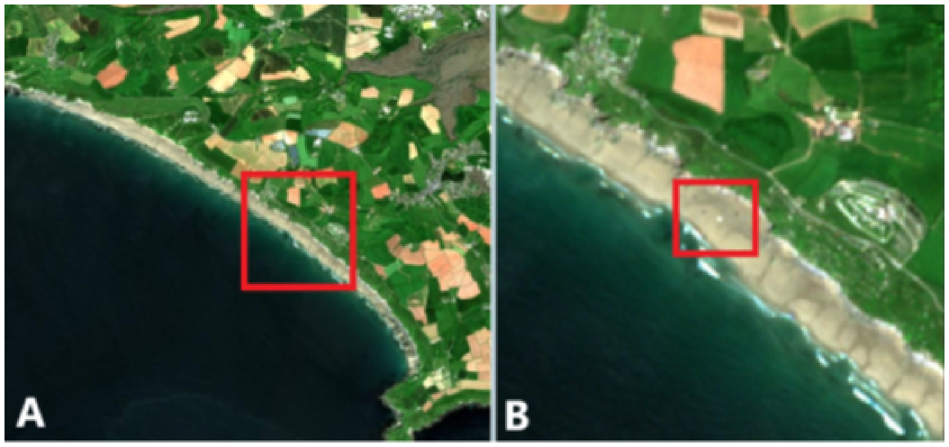


Figure 2.1: Example of a plastic debris detection experiment during a Sentinel-2B overpass on 15 May 2018 over Whitsand Bay (United Kingdom). (A) Overview of the bay area with the study area indicated by a red square, shown in detail in (B) with the positions of the plastic targets (10 × 10 m) visible within the red square (Martínez-Vicente et al., 2019).

### 2.2.2 Unmanned aerial remote sensing

Unmanned Aerial Vehicles (UAVs) are powerful tools to acquire low altitude remote sensing data in order to obtain a synoptic overview of extended areas. The use of an automated flight path as a monitoring technique provides an easy, adaptable and relatively cheap method to monitor litter items either on beaches or at locations without existing infrastructure.

Several kinds of sensors can be embedded to an unmanned vehicle to operate in different environments and conditions with different levels of autonomy. Inertial sensors like accelerometers and gyroscopes combined with a magnetometer are used to determine the flight position and orientation of an unmanned vehicle. Barometric pressure sensors are used to determine the height. The information of these sensors is provided to the flight controller, which monitors and controls everything the UAV does. For trash detection, Kraft et al. (2021) used a commercial flight controller, which encompasses a combination of accelerometers, gyroscopes, and barometers to provide relative position data, and it is connected to a Global Positioning System (GPS)/Global Navigation Satellite System (GNNS) module to provide absolute positioning information. The optical sensor mounted on a UAV for applications like tracking litter items is an RGB camera to acquire high resolution images that can be used along with the other sensors to produce high quality aerial imagery and digital surface models.

Compared to the use of piloted aircraft or satellite imagery, an unmanned aerial vehicle operates at low altitude, thus allowing the acquisition of high spatial resolution imagery. Generally, UAVs are low-cost platforms offering a good solution to be used more frequently. Flight planning is easy since several software solutions are developed to control autonomous flights. The spatial resolution achieved in the collected imagery allows integration with object detection algorithms, which could drastically reduce observational errors, and therefore, lead to more accurate estimations of trash recognition. Nevertheless, UAVs should not be used when the wind velocity exceeds 10 m/s and during rain, snow, or thunderstorms, because the altitude accuracy will decrease radically (Geraeds et al., 2019), the battery life will be drained faster, and the electronic gear can wear down.

There is similar research focused on the development of a UAV-based protocol or a combination of protocols to automatically detect and quantify beach litter (Fallati et al., 2019; Gonçalves et al., 2020) and floating riverine plastic debris (Geraeds et al., 2019). For study area surveying, the researchers used commercial quad-rotor UAVs, equipped with RGB high-resolution cameras, that can smoothly fly at low altitude to obtain good ground-resolution images compared to fixed-wing UAVs. Fallati et al. (2019), selected the UAV altitude of 10m, to obtain a spatial resolution expressed in ground sample distance (GSD) of about 4.4 mm/pixels, given by the equation below:

$$GSD = \frac{SW \cdot FH}{FL \cdot IW} \tag{2.1}$$

where $SW$ is the sensor width, $FH$ is the flight high, $FL$ is the focal length of the camera, and $IW$ is the image width (Ventura et al., 2018). Regarding autonomy, coverage and image resolution, Gonçalves et al. (2020) found best to fly the UAV at an altitude of 20m, corresponding to a GSD of about 5.5mm/pixels, suitable for detecting meso-litter items (size between 2.5 cm and 50 cm). Furthermore, Lo et al. (2020) conducted UAV flights at different operating heights and light conditions, concluding that the UAV is best to be flown at 5m-10m operating heights on a sunny day for litter items not smaller than 10 cm.

To study the spatial and temporal accumulation dynamics of beach litter, Merlino et al. (2020) involve unmanned aerial remote sensing as it provides good repeatability in surveying by setting pre-defined flight parameters (flight altitude, flight starting-ending point, flight time, etc). Their results indicated that the accumulation dynamics of beach litter depends not only on the season, but also on the size of litter and on extreme weather events. Moreover, they compared standard monitoring campaign results with the results obtained from aerial images, regarding the size and classification of beach litter. The comparison showed good agreement for medium and large size objects ($\sim$67%-95%), but not for small ones ($\sim$15%).

Regarding the hardware involved in unmanned aerial remote sensing, Kraft et al. (2021) developed a custom-made sensor system based on deep learning object detection able to operate onboard the UAV and analyze litter imagery in real-time. As stated previously, for flight control and positioning, they used a commercial autopilot (Pixhawk), which integrates the necessary, and some redundant, inertial measurement sensors, and is externally connected with a GPS/GNSS module. Multiple embedded computational platforms were tested in a wide range of configurations for visual information processing, since deep neural networks are computationally expensive in terms of processing power, time and memory. The researchers report, based on their results, that the most reliable platform is the NVIDIA Jetson Xavier NX module (NVIDIA Developer, 2020) because of the range of solutions it can run. When running 16-bit floating point version of YOLOv4 object detection algorithm (Bochkovskiy et al., 2020), it exhibited near real-time performance having the capability to process more images per second with good detection accuracy compared to other configurations.

Although unmanned aerial systems are extremely versatile and useful tools for the investigation and analysis of a number of environmental issues, they have their limitations. These systems can be deployed when atmospheric (i.e., cloud-free), environmental, and solar conditions are acceptable to study specific phenomenon. They are not ideal for covering large areas (not enough battery life, high cost per unit area of ground coverage). Additionally, aerial missions are often carried out as one-time operations, whereas earth observation satellites offer the possibility of continuous monitoring of the earth. Therefore, if the goal is to map areas with large extents in different resolutions, satellite imagery is the appropriate choice. If the research aims to detect small marine features, high resolution data provided by aerial systems are required. In some cases, including the tracking of marine debris, satellite and airborne sensors are needed to be utilized.

## 2.3 Litter detection

Object detection technology aims to identify target objects in collected images/videos and determine the category each one of the target objects belong. But before the detection, data has to be collected and preprocessed with methods that will serve the end-purpose of the problem. Afterwards, to be able to infer from the collected data, object detection models have to be trained and evaluated. This section discusses about data collection-preprocessing methods applied for litter monitoring, the current training model methods as well as algorithms that have been used in the latest research for litter detection.

### 2.3.1 Data preprocessing

When collecting imagery either on ground level or in aerial mode, researchers tend to follow a SfM (Structure from Motion) photogrammetric pipeline to obtain a 3D reconstruction of the target objects (Fallati et al., 2019; Gonçalves et al., 2020; Merlino et al., 2020; Lo et al., 2020). To conduct this kind of workflow, photos of the target objects should be taken such that there is an overlap between the adjacent frames with a coverage of about 70% to 80%. The tool used by the researchers to perform photogrammetric processing is the Agisoft Metashape, or formerly known as Agisoft Photoscan. In aerial mode, the SfM processing can be used to define the position, shape and size of the objects on the ground. The acquired images are loaded to the software to create an orthomosaic for the entire study area. Figure 2.2 depicts the photogrammetric technique that allows the generation of 3D spatial data. Specifically, Merlino et al. (2020) state *"By precisely knowing the position of the homologous points A' and A" on the two photographs, and the spatial position of the* *two sectors and the two perspective centers O1 and O2, the point A remains geometrically defined, since it is the intersection point of the two projecting rays r1 and r2 connecting the two homologous points with the perspective centers. This does not happen with a single photo shoot".*

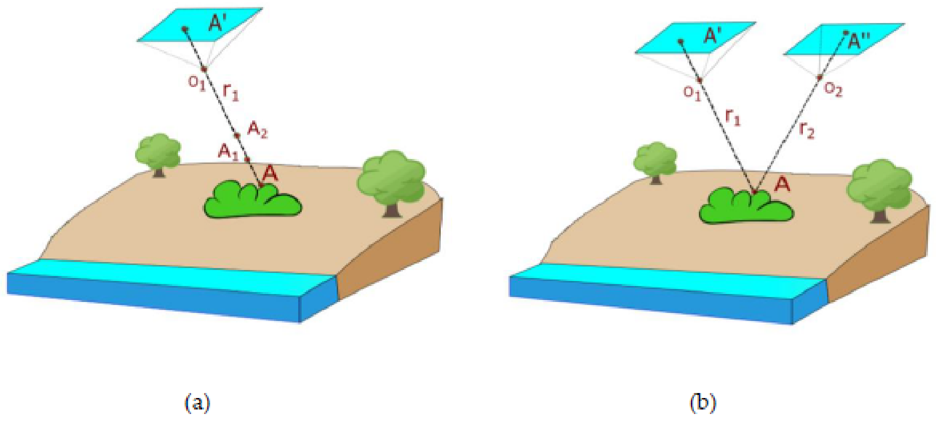


Figure 2.2: Representation of photogrammetry technique demonstrating triangulation to generate 3D spatial data (b) and comparison with a single photo shoot (a) (Merlino et al., 2020).

For monitoring and detecting litter on the streets, Ping et al. (2020), located a mobile device on a trash truck to capture images of streets. Three cameras were placed to look in different direction to take high-resolution pictures. The acquired pictures were passed into an edge processing component for preprocessing using a limited computing power. The preprocessing involves checking the clearness of the captured images, and applying an optimized pre-trained deep learning model to determine any regions of interest in images which will be used for object detection in the subsequent processing. Candidate images for further processing, are compressed and encrypted before being transmitted to the cloud server for object detection.

### 2.3.2 Algorithm training and datasets for litter detection

Before referring to the algorithms used for litter detection in the literature, it is necessary to discuss about the deep learning model training used for object detection. Deep learning models learn by providing them with data and letting them find progressively the relationships that represent the data best. The data given to the model is labeled. For example, the images provided to a deep learning model for object detection are annotated with a bounding box for each item they contain and a corresponding label. For image segmentation purposes, they can also contain a segmentation mask for each detected object. Thus, the learning algorithm trains itself by trying to develop understanding of the patterns in order to be able to detect and map an object to its label. The data provided to a learning algorithm is called training data and the better the quality and quantity of this data is, the better the model performs.

The amount of data provided to a deep learning algorithm plays a significant role on its performance. Generally, this kind of algorithm needs a large amount of training data to perform successfully. The number of training data required depends on the complexity and seriousness of the case. For more complex problems, the deep learning model needs more training data as a rule of thumb. More serious problems require a model that is more confident on its results, hence, require as many data as possible. The number of object classes the model has to identify is also an indicator of the number of training examples to be used.

The nature of each problem raises the need for field-specific training datasets. There is a number of open-sourced datasets used for algorithm training and evaluation. For trash detection and segmentation, there is the TACO (Trash Annotations in Context) dataset by Proença and Simões (2020), that contains litter imagery taken under diverse environments. For underwater object detection and segmentation, there is the TrashCan dataset by Hong et al. (2020b) which contains observations of trash, a variety of undersea flora and fauna, and unknown objects. For the classification task of litter items, there is the Trashnet dataset (Thung, 2016) which includes garbage photographed in a clear background, and the WaDaBa dataset (Bobulski and Piątkowski, 2018) which consists of plastic waste also photographed in a clear background. Recently, a dataset consisted of drone imagery used for litter detection, called UAVVaste, was developed by Kraft et al. (2021). The reason of its development is that the object detection algorithm has to be trained to detect a relatively small size of objects in the image, which is more difficult to achieve with a regular trash dataset.

Apart from ready-to-use training datasets, researchers can apply techniques like data augmentation to generate an effective training dataset for object detection models. Data augmentation is applied to enlarge an image dataset, by making minor alterations to the existing images, such as changing the orientation, scale, brightness, etc. Advanced techniques of producing a training dataset involve deep generative models, which are deep neural network architectures with an incredible ability to generate highly realistic data of various kinds such as images, texts or sounds to expand the original dataset (Ruthotto and Haber, 2021). Hong et al. (2020a) produced a synthetic dataset of realistic images of underwater thrash using a major family of deep generative models, called Variational Autoencoders (VAEs). VAEs encode input data as distributions constrained to be close to a standard Gaussian. In this way, when sampling data from these distributions, interpretability will be ensured and closely mapped data will return similar content. Another example of expanding a training dataset comes from Olivelli and Rosebrock (2020). They used a synthetic dataset of trash images created by Microsoft's synthetics team based in Seattle. Various objects were superimposed over backgrounds obtained from their field photos, managing to produce thousands of synthetic images over a relatively small time interval.

When training a deep learning object detection model, there is a procedure that is very popular and often followed by research community. This procedure involves the use of a model pre-trained on related data, because it can help train, more quickly and efficiently, deep learning models with comparatively little data. This idea focuses on using the knowledge gained from solving a related task and applying it to improve a model's generalization in another task. In particular, the parameters of the pre-trained neural network, where the knowledge of solving a relevant problem is stored, are transferred to the new network. This is the reason why this procedure is called transfer learning. Typically, most problems do not have a large dataset of labeled data points to train such complex models as deep learning models. For this reason, transfer learning gains in popularity. Regarding trash detection, researchers often use models pre-trained on the Microsoft's COCO (Microsoft Common Objects in Context) dataset (Lin et al., 2015), to take advantage of transfer learning (Fulton et al., 2019; Ping et al., 2020; Kraft et al., 2021). COCO dataset contains images with over 1.5 million objects instances people encounter on a daily basis and image annotations for 80 object categories, and is available for public use. It is suitable for litter items detection because it includes small objects in various backgrounds. Therefore, it improves generalization and reduces the chances of overfitting.

### 2.3.3 Object detection algorithms for litter detection

In literature, the algorithms used for detecting trash vary in their type and complexity. Gonçalves et al. (2020) conducted object-based image analysis through a commercial software. They applied three popular machine learning classifiers, Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Random Forest, to create litter abundance maps. Their study case includes monitoring litter items in sandy beaches with a consumer-grade UAV. They measured the precision and sensitivity of each algorithm and concluded that for a multi-class detection, the Random Forest algorithm, which is the most complex of the three regarding optimization, had the highest performance.

More research on tracking litter items focuses on deep learning-based object detectors. Fallati et al. (2019) applied a commercial deep learning software, called PlasticFinder, to detect and quantify marine debris on shorelines from aerial images. Prior to using the software, they performed a comparison between the number and the type of the marine litter counted during a ground assessment on the study area and the number and the type of the marine litter counted via image screening on the PC. The images that were considered suitable to train and test PlasticFinder had a matching score over 80%. When compared to a previous similar research by Martin et al. (2018), which used multi-class Random Forest classifiers, the PlasticFinder scored much higher precision.

Due to the introduction of convolutional neural networks (CNNs), object detection in images has seen enormous progress in terms of accuracy and speed. CNN-based object detection algorithms, such as Faster R-CNN (Ren et al., 2017), Mask R-CNN (He et al., 2017), YOLOv3 (Redmon and Farhadi, 2018), EfficientDet (Tan et al., 2019), are used extensively by the research community. Ping et al. (2020) trained a Faster R-CNN model to detect and classify litter on images collected by various public image sources. First, the training occurred for a small dataset with 6 litter classes, and for the performance evaluation of the model they used the PASCAL VOC metric (mean average precision for an IoU threshold equal to 0.5) that is discussed in section 3.6.4. Second, the training took place for a large dataset with 11 classes of litter. The model was evaluated with the COCO dataset metrics, also discussed in section 3.6.4. In the training procedure of the Faster R-CNN model, instead of using random weight values, they applied the weight values of another Faster R-CNN model trained on the COCO dataset for the purpose of transfer learning. On a similar theme, Kraft et al. (2021) compared the performance of real-time object detection algorithms, like YOLO and EfficientDet variants, pre-trained on the COCO dataset and trained on the UAVVaste dataset for trash detection on aerial images.

## 2.4 Conclusions

Currently, the most common method for monitoring trash relies on ground assessments. This process has the advantage of an easier and more reliable classification compared to remote sensing methods because of the direct encounter with trash. Nevertheless, this process in nature is labor-intensive because of the difficulty of surveying many locations, sometimes of great extent, and over extended periods.

Remote sensing surveys are the simplest sampling technique, in which all analytical methods can be applied in order to deduce garbage properties, including composition and possibly even origin. A satellite remote sensing system can provide global scope observations, and continuous temporal coverage, quantitative measurement of ground features, and semi automated computerized processing and analysis. However, remote surveying by satellites has its limits regarding the acquisition of detailed spatial information. Unmanned aerial systems can be used instead of satellite systems or as supplementary tools depending on the study case. They provide high spatial resolution images, thus making easier the detection and classification of waste. The pixel densities they provide, allow for integration with object detection algorithms, which can limit observational errors significantly. But UAVs should not be used in severe weather conditions as their efficiency will decrease substantially, because of the possibility their gear will get damaged.

The customization of object detection algorithms for the identification of trash items on influenced areas has been proven to be a beneficial tool in current and future research. But first one has to pay attention to the way data is collected and preprocessed, because the quality and quantity of the data affects the performance of object detection models. In general, the training of a deep learning model on images requires good spatial resolution and a lot of image data in order for the model to infer better. To have enough data to train a deep learning object detection model for a certain study case, researchers often use a large dataset which is related to their problem. There are several datasets available for training-testing object detection algorithms. These datasets are also used for pre-training a model, before applying custom relevant data to it. In this way, the knowledge from the pre-trained data will be transferred to the deep learning model, and when the model is exposed to the smaller dataset will be able to generalize more and not overfitting to the data.

Current research on litter detection shows a tendency towards the use of deep learning-based object detection algorithms. Most of it, focuses on applying a detection algorithm on images acquired from a UAV, usually aiming either to automate completely the procedure of detection/classification of litter items on coastal environments, or to compare the algorithms results with the results of on ground assessments. Depending on their level of expertise in object detection algorithms, researchers have used from simple machine learning classifiers and commercial deep learning-based software to complex architectures that incorporate convolutional neural networks. Whatever the type of object detection algorithm applied, the common goal is to obtain as accurate results as possible.

# Chapter 3

# Theoretical Background

## 3.1 Introduction

Image classification and object detection are part of the computer vision area, which aims to reproduce the capability of human vision. Image classification is the task of assigning a label to an input image from a fixed set of categories. Object detection is the task of locating one or more objects in an image and classifying it/them by assigning a label from a fixed set of categories. It is evident from the definitions that image classification is a simpler task to perform, and that object detection encompasses the image classification task. These computer vision tasks share a common goal, which is to understand the content of images by extracting information from them. They take images as input and give output in the form of information. Due to the advances of deep learning techniques, convolutional neural networks, and the increase of the parallel processing power offered by the graphics processing units (GPUs), these computer vision tasks have been rapidly developed in the last years (Chollet, 2017). There are plenty of practical applications involving image classification or object detection such as computer-aided medical diagnosis and treatment planning (Robertson et al., 2018; Traore et al., 2018; Ulhaq et al., 2020), plant diseases diagnosis (Dhingra et al., 2019), mineral grains recognition (Maitre et al., 2019), nuclear and particle physics applications to the analysis of Large Hadron Collider events

(Schwartzman et al., 2016), fraud events detection for financial security matters (Sahni et al., 2020).

With the application of computer vision technology, calculations and testing have become more accurate and today's systems have evolved to react to visual inputs quicker than humans. This chapter intends to present an extensive study background to provide the reader with the basic theory of concepts closely relevant to this study, like deep learning, convolutional neural networks, object detection algorithms and popular evaluation metrics.

## 3.2 Machine learning

In programming, problems are often approached with logical and methodical thinking. Depending on the desired output, the appropriate rules are set in a way that will convert the input into the output. In machine learning, the program itself learns the rules that best describe the data, gradually uncovering patterns in the data with each iteration. As more patterns are revealed, its performance becomes better and better.

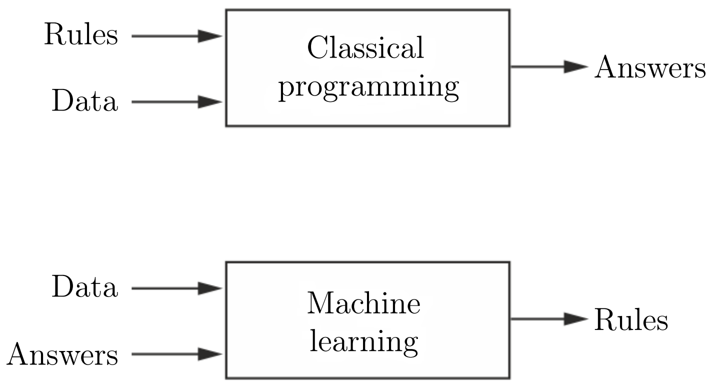


Figure 3.1: Machine learning vs classical programming (Chollet, 2017, chapter 1).

Although machine learning was initially avoided due to its high computational requirements and the limited computing power at the time, it began to flourish in the last several years due to the preponderance of data arising from the rapid increase of information as well as the development of graphics processing units (GPUs).

Machine learning is closely linked to statistics, as it involves learning algorithms, the properties of which are studied using a statistical framework. It is based on statistical learning theory through the process of inductive inference, that is:

1. making observations,
2. discerning a pattern,
3. making a generalization, and
4. infer an explanation or a theory (Bousquet et al., 2004).

It aims to automate this process by searching for hidden trends that describe the input data within a predefined space of possibilities for a given task. Then, it creates a model that explains some real-world data and makes predictions on what may happen next with different inputs. But unlike statistics, machine learning (especially deep learning) is engineering oriented as ideas are proven more often empirically than theoretically (Chollet, 2017).

The machine learning algorithms make mathematical models based on the given data to make predictions. The data that goes into a machine learning model consists of measurable properties, called the features, which have to be carefully selected or "engineered" for the model to be functional. The process of selecting the features in machine learning model implementations to make them functional is called feature engineering. The functionality of a machine learning model is determined by four properties:

- Performance - how well it performed on the given data,
- Runtime - how fast it runs,

- Interpretability - how comprehendible its predicting operation is,
- Generalizability - how well it generalizes to unseen data.

Researchers have to dedicate time to prepare and optimize their machine learning models according to the aforementioned properties to make sure of their functionality. On the other hand, this is not the case with deep learning, where feature engineering is essentially automatic. Deep learning algorithms need only little human intervention. They are strongly characterized by their self-learning capabilities and their ability to transform the given data into an abstract representation.

## 3.3 Deep learning

The term learning in machine learning refers to the process by which models find the rules that describe the data. A machine learning algorithm uses these rules to find the best representation of the data involved. Then, the algorithm uses this data representation to make predictions about new data that it has never encountered before (Goodfellow et al., 2016).

Deep learning is a subfield of machine learning and has a very similar process of learning:

- build a model
- add relevant data,
- train model on the relevant data,
- test the predictive power of the model on data it has never encountered.

The term deep in deep learning concerns the numerous data transformation layers. At each successive layer, the deep learning algorithm processes the input data to learn progressively complex patterns. This is an architecture that mimics the structure of the brain.

### 3.3.1 Artificial Neural Networks

Deep learning models are biologically inspired. They are an attempt to simulate the way the human brain works and finds patterns for decision making. Deep learning integrates artificial neural networks into its architecture. An artificial neural network consists of:

- *Neurons or nodes* - An artificial neuron is a mathematical function designed to imitate the functioning of a biological neuron. It can take an input, process it based on some rules and output a result. Then, it trains itself on its own result and gives better results in the future. In other words, it learns by trial and error just like a biological neuron. With mathematical terms, it computes the weighted average of the data input and passes the information through a non-linear function, called the activation function (CS231n by Stanford University, nd).

- *Connections between the neurons/nodes* - Each connection has a positive or negative value on it, called weight, that controls the strength of influence of one neuron on another.

- *Layers* - A neural network can have multiple layers of neurons. The first layer is the input layer, where inputs enter the neural network. The final layer is the output layer and produces the final result. There are also layers in between the input and output, called hidden layers (Goodfellow et al., 2016, chapter 6). These layers introduce complexity into a neural network and generally aid in the learning process. Depending on the number of hidden layers (which can be zero), a neural network can be characterized as shallow or deep.

Figure 3.2 shows an illustration of a human neuron (a) versus its mathematical model (b). A human neuron consists of dendrites, nucleus, cell body and axon. Impulses are received by dendrites, processed to be information in the cell body and the output is sent along its axon and via the synapses to the dendrites of the next neuron. In an artificial neuron, the impulse is an input signal, $x_0$, that travels along the axon of the previous neuron and interacts with the dendrites of the neuron based on the synaptic strength of a particular synapse, $w_0$ (CS231n by Stanford University, nd). The synaptic strength is not a static quantity and can be modified. Likewise, in an artificial neuron the synaptic strength, referred to as weights, is a learnable parameter that strengthens or weakens the input signal and can be modified to output a better result. The weighted input signals ($w_i \cdot x_i$) are carried to the cell body where they get summed. A bias is also added to the weighted sum in order to increase accuracy of the output, acting like the constant of a linear function. A biological neuron produces a signal along its axon when it is stimulated or activated by other neurons to which it is connected. For an artificial neuron, the activation is done with the use of an activation function. The value of the final weighted sum can be anything ranging from $-\infty$ to $+\infty$. The activation function sets a certain threshold to the value of the final sum. If the final weighted sum is above the threshold, then the neuron is activated and outputs a signal.

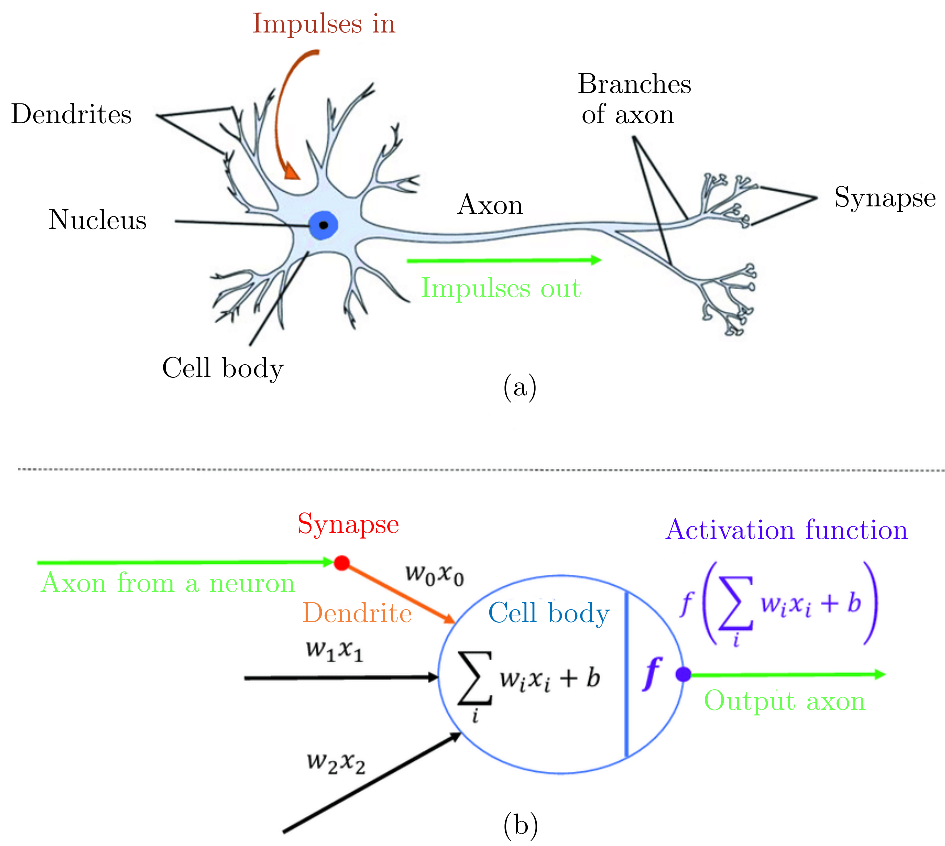


Figure 3.2: (a) Human neuron (b) Artificial neuron (CS231n by Stanford University, nd).

The biological neurons are complex nonlinear dynamical systems. Thus, the activation function is nonlinear. Non linearity creates more complexity during the learning process and gives the model the power to learn non-linear patterns in the data. If a neural network had many layers performing only linear transformations, it would simply be a series of successive layers that would not be more efficient or accurate than simple linear regression.

There are various types of activation functions that can be applied at each layer of the network. The sigmoid activation function is commonly used in the final layer of the network and for binary classification problems as it produces values in the range [0,1]. The sigmoid function is defined as:

$$A = \frac{1}{1+e^{-x}} \tag{3.1}$$

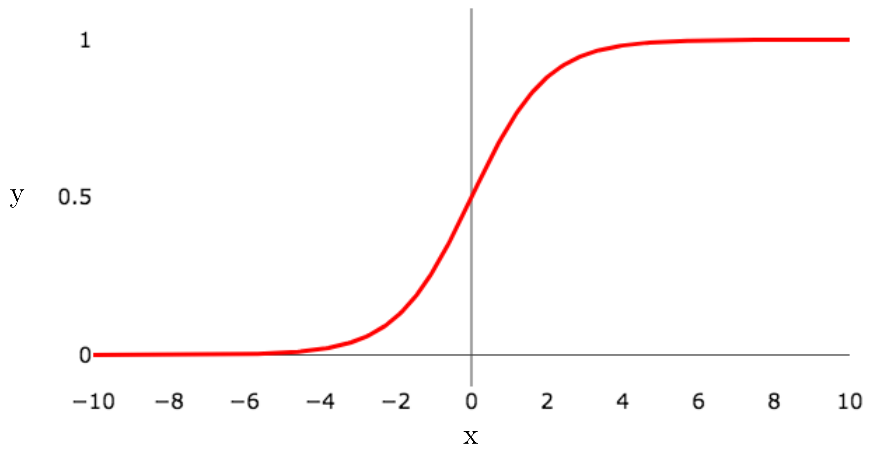


Figure 3.3: Sigmoid activation function (Cooper, 2018).

A general problem of this activation function is that towards either end of the curve, the y values tend to respond very less to changes with x. The gradient is very small near the horizontal parts and hence the neural network becomes drastically slow in learning.

The most popular activation function for hidden layers is the Rectified Linear Unit (ReLU) (Goodfellow et al., 2016, chapter 6). The ReLU function, shown in Figure 3.4, gives an output x if x is positive and 0 otherwise. It can be defined as:

$$A(x) = max(0, x) \tag{3.2}$$

The benefits of this function are the following:

- *It is easier computationally* - There is no calculation of exponent in this function.

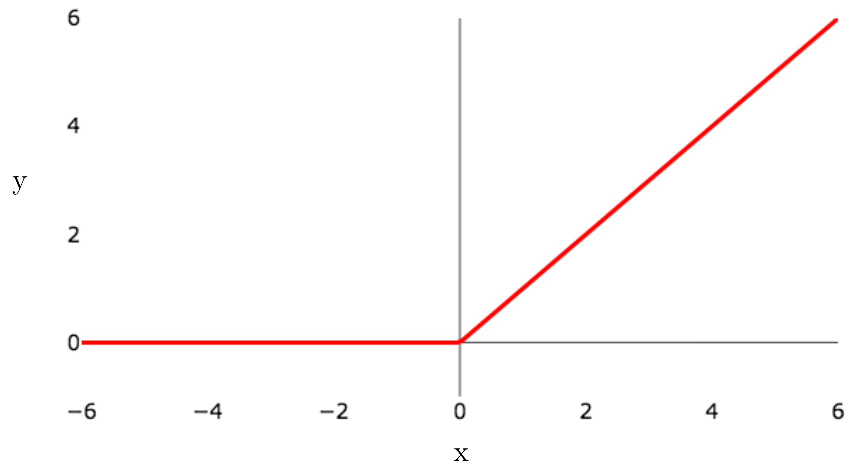


Figure 3.4: ReLu activation function (Cooper, 2018).

- *It makes the network lighter* - Because negative and zero inputs produce zero output, there are less activated neurons in the network creating a sparse representation, which is a property that makes the neural network more efficient.

- *The linear part of the function solves the problem of the vanishing gradients*(a problem encountered when using other functions like the sigmoid) - The y values change proportionally with the x values not hindering the learning process.

However, the range of ReLU is [0, inf], meaning the activation of the neurons is not bound to a certain limit. Also, for negative values there will be no activation of the neurons, thus the network will be partially passive in learning. This issue can be resolved by using variations of the ReLU by simply making the horizontal line into a non-horizontal component.

In each of the layers of a neural network, neurons accept an input, process it linearly by calculating the weighted sum and based on this sum, they "decide" whether to pass the information further or not (they get activated or not). If they get activated, the information is passed to the activation function and the result value is fed as input to the neurons of the next layer. This process happens in the forward direction meaning the input data are fed to the input layer, passing through each successive layer until they reach the final layer of the neural network. This process is called forward propagation.

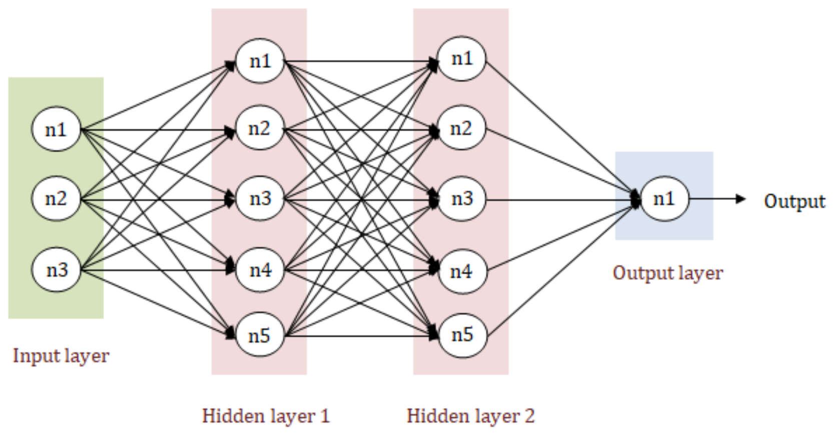


Figure 3.5: Neural network with two hidden layers.

After inputs have gone through a neural network one time, the output is unlikely to be accurate. The output is used to estimate the network's prediction error with a loss function. The loss function is used to calculate the error between the predicted values and the actual values (Goodfellow et al., 2016, chapter 6). For classification learning models, where the aim is to map the input variables to a class label, the model predicts the probability of an example belonging to each class. The loss function measures how much the predicted probability diverges from the actual label. This type of loss function is called cross-entropy loss.

In order to produce an accurate output, the model has to operate in a backward way, from the output layer to the hidden layers, to adjust its learning parameters, i.e. the weights ($w$) and bias ($b$), in a way that the error will be decreased. The model uses an algorithm, called backpropagation, to calculate the gradient of each learning parameter, working backward, propagating the error in the predicted output. Then, an optimization algorithm, called gradient descent, uses the gradient of each parameter to update $w$ and $b$ in a way that the loss function $L(w, b)$ will be minimized (Goodfellow et al., 2016, chapter 6). This process of updating the weights/bias between the neurons is done iteratively until the desired output is achieved. The direction and magnitude by which the learning parameters are updated is called the error gradient and it is estimated with the loss function. Figure 3.6 depicts the gradient descent's work of finding the local minimum of the loss function $L(w, b)$ in order to increase the accuracy of the model. The variable $\theta$ is the model parameter and $\xi$ is the optimal value of the model parameter (Chen et al., 2020).

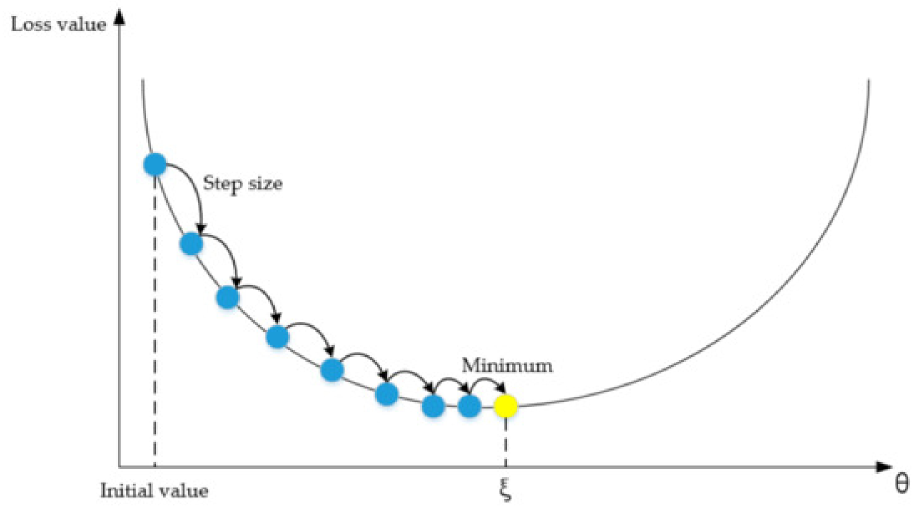


Figure 3.6: Schematic representation of the gradient descent operation (Chen et al., 2020).

The step size the algorithm takes into the direction of the local minimum is controlled by the learning rate of the model. The learning rate is a hyperparameter that refers to how much the parameters are changed on each iteration. Equation 3.3 describes what the gradient descent does: the gradient $\nabla f(\theta)$ is multiplied by the learning rate, $\alpha$, and the product is subtracted from the current parameter in order to find the next $\theta$.

$$\theta_{new} = \theta - \alpha \cdot \nabla f(\theta) \tag{3.3}$$

For gradient descent to reach the local minimum the learning rate must be set to an appropriate value. If the value is too big the algorithm will never reach the optimum and if it is too small it will take too much time to achieve the desired value.

The main components of a neural network learning pipeline is depicted in the following scheme. For classification tasks, the input data that is used to train the network are features and true labels for each feature. The model predicts for each data example what its label should be. The output is used to estimate the error based on the true labels of the training data. The optimizer based on the information of the loss function adjusts the parameters of the network in the right direction.

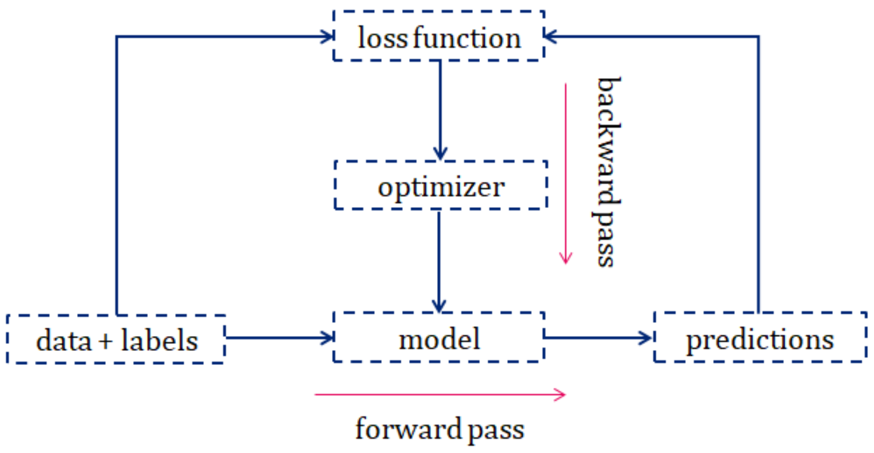


Figure 3.7: Neural network learning pipeline.

Neural networks are parametric models meaning they are described by configuration variables representing the model's knowledge. These parameters can be adjusted using the training data. Then, the performance of the model will be evaluated using hold-out test data the model has not seen during training. While the model parameters are the ones that the model uses to make predictions, there are also parameters called hyperparameters that determine the learning process. Hyperparameters are not estimated from the data and they are set and modified manually in order to help estimate the model's parameters. Some of the hyperparameters a deep learning model has are the:

- *Learning rate*: refers to how much the parameters are changed on each iteration.
- *Batch size*: determines how many training samples are used before updating the network's parameters.
- *Number of epochs*: the number of times the whole training data is used by the network while training.
- *Number of hidden layers*: the number of the network's intermediate layers.
- *Optimizer*: the algorithm that based on the training data adjusts the parameters of the network.

Hyperparameters are set before training and they are tuned after training using a held-out dataset called validation test. Using the training dataset to determine the hyperparameters might lead to overfitting to the training data. Overfitting means that the model learns patterns specific to the training data that would not apply to new data.

Almost any deep learning project will have the pipeline shown in the figure below. Data is split into a:

- training set to train the model by updating the parameters (weights and bias) of the neural network,
- validation set to evaluate the model's training. It is a biased evaluation because the model is modified based on its performance on this set,
- test set for an unbiased evaluation of the model.

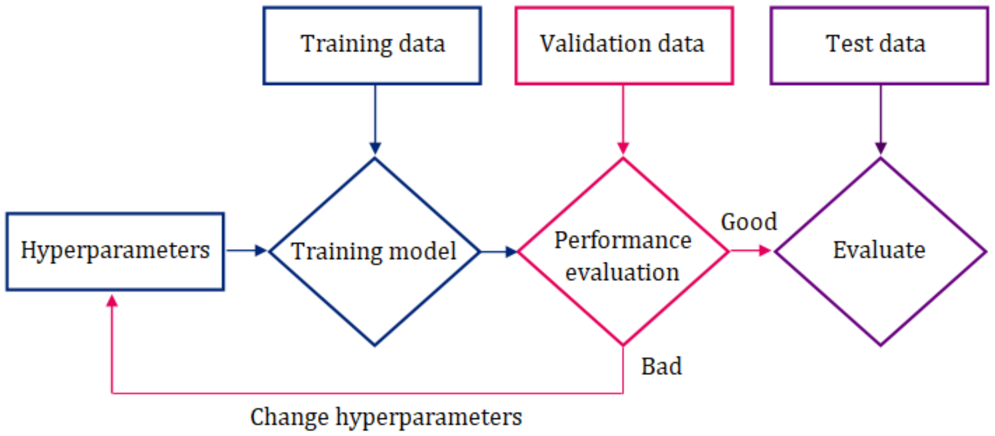


Figure 3.8: Deep learning model workflow.

## 3.4 Convolutional Neural Networks

Just like any machine learning method, a neural network learns how to perform tasks by processing data and adjusting its model to best predict the desired outcome. Neural networks are very effective for high-dimensionality problems and they are able to model complex relationships between variables. They are often used for image classification and object detection because they seek to find the complex patterns necessary to map an image or object to its label. To implement a classification task, deep neural networks involve layers designed to process image data by focusing on local relationships between features. These layers, called convolutional layers, scale down the input images into meaningful features while using only a fraction of the parameters required in a linear layer. The neural networks that use these types of layers are very similar to ordinary neural networks and they are called convolutional neural networks.

### 3.4.1 Convolution

In deep learning models, convolutional neural networks, or CNNs, are specifically designed for processing data that has a known grid-like topology, like image data (Goodfellow et al., 2016, chapter 9). The role of the CNN is to transform the images into a shape that facilitates processing without losing information, which is important for an accurate prediction. This is critical to designing a model that is not only good at learning, but can also scale to huge datasets.

CNNs use a mathematical linear operation called convolution in at least one of their layers, instead of traditional matrix multiplication (Goodfellow et al., 2016, chapter 9). A 2-D convolution is mathematically represented as:

$$g(i,j) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h[m,n] \cdot f[i-m, j-n] \tag{3.4}$$

Since convolution is typically denoted with an asterisk, the formula can be simply written as:

$$g(x,y) = h(m,n) * f(x,y) \tag{3.5}$$

where $h(m,n)$ is the input image matrix, $f(x,y)$ it the kernel, or filter, which is a tensor containing parameters/weights that are adapted by the learning algorithm, and $g(x,y)$ is the output, which is often referred to as the feature map.

Convolution is equivalent to flipping the kernel in both dimensions, horizontally and vertically, and applying cross-correlation, also known as sliding inner-product. Conceptually, this means that the flipped kernel "slides" onto the image matrix and the new pixel values are estimated by the inner product of the overlapping pixel values with their corresponding filter weights.

There are plenty of machine learning libraries that implement cross-correlation, without kernel inversion and call it convolution. For symmetric kernels, convolution and cross-correlation result in the same output.

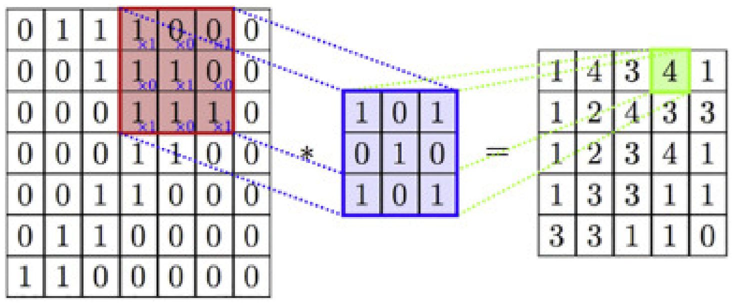


Figure 3.9: Image processing convolution (Traore et al., 2018).

The mathematical formulation of 2-D cross-correlation is given by:

$$g(i,j) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h[m,n] \cdot f[i+m, j+n] \tag{3.6}$$

more simply represented as,

$$g(x,y) = h(m,n) \otimes f(x,y) \tag{3.7}$$

In convolutional networks, the kernel matrix is typically smaller than the input in order to be able to detect small, meaningful features such as edges. On one hand, a set of smaller weight tensors (kernels) means less parameters to store, which reduces the memory requirements of the model and improves its statistical efficiency (Goodfellow et al., 2016, chapter 9). On the other hand, a small tensor results in fewer operations when computing the output.

The systematic application of the same kernel across an image is the innovation of convolutional neural networks. The kernel can detect specific features in the input image, depending on its design. Therefore, its systematic application across the entire input can enable the detection of those features anywhere in the image.

### 3.4.2 Architecture

There are various architectures of convolutional neural networks available which follow similar design principles. These architectures are key in building algorithms which machine learning practitioners will then adapt to solve various computer vision tasks. Some classic network architectures are, namely, the AlexNet (Krizhevsky et al., 2012), VGGNet (Simonyan and Zisserman, 2015), Inception or GoogLeNet (Szegedy et al., 2015), ResNet (He et al., 2016). These deep learning networks are often used as feature extractors for the object detection task.

### Convolutional layer

Convolution takes place in the convolutional layers of the CNN. The convolutional layer is always the first hidden layer of a CNN and contains a set of units/neurons. Also, these layers contain a predefined number of filters/kernels with a predefined size. The weight values within the kernels are the learnable parameters during the training phase of a CNN. They are randomly initialized and then they are learned via backpropagation.

Each neuron of the convolutional layer is exposed to a defined region of the input data, called the local receptive field, rather than the whole input (Chen et al., 2014). This receptive field has the size of the defined kernel within the layer. Therefore, the size of the kernel indicates the extent of input data each neuron is exposed to. Each neuron is responsible for extracting features of the input that are in the receptive field which may contain patterns such as lines and edges or small details that make up the image. The exposure of a neuron to a defined region of the input instead of the whole input makes the convolutional neural network more computationally efficient since it processes fewer parameters and hence computes the output with fewer operations.

The hyperparameters for a convolutional layer are the:

- *Number* and *size* of the kernels applied in the input data.
- *Stride*: determines how much the kernel is moved each time it is applied.

- *Padding*: defines what is done once the kernel gets to the end of a row/column of the input matrix. The default option is to stop when the kernel moves off the image. That is referred to as *valid padding*. Another option is to pad the input by surrounding it with zeros. This option is called *same padding*.

Figure 3.10 depicts on the left side a 227×227 input image and the convolutional layer's receptive field of 5×5. Each neuron of the convolutional layer is connected to 5×5, which is 25, weights. The output dimension of the convolutional layer has a depth component which corresponds to the number of feature maps that are created. The number of feature maps are determined by the number of kernels used in the convolutional layer, which is 56 in the example below.

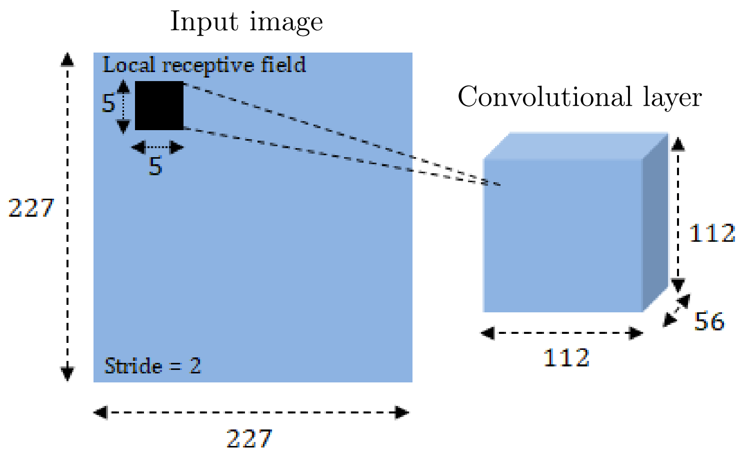


Figure 3.10: Illustration of a convolutional layer and its local receptive field.

The convolutional layer's width dimension is estimated by Equation 3.8:

$$layer\_width = \frac{W - w + 2 \cdot padding}{stride} + 1 \tag{3.8}$$

where $W$ is the image width, $w$ is the kernel width, and 1 unit representing the bias. The convolutional layer height dimension can be calculated by substituting the width in equation 3.8 with the height dimension of the input image and the kernel.

A feature map is the output of a kernel convolving the input matrix (with a stride of 2 and valid padding in the example). So the neurons of the feature map are connected with the same weights. That means that the network learns one set of parameters per feature map. Thus, the number of trainable weights of the convolutional layer is defined by the number of feature maps or kernels multiplied by the number of weights each kernel has (plus the bias). In the example above it will be 56×(5×5+1) or 1,456 trainable parameters. Learning only one set of parameters per feature map is a powerful characteristic of CNN. It is referred to as parameter or weight sharing because the neurons of a feature map are sharing the same weights. One advantage of this is that the neurons of a feature map are trained to detect the same features on the input image. If the features detected are vertical lines, then the feature map will contain only the vertical lines of the input image. Another advantage of parameter sharing is the drastic reduction of the layers' trainable parameters. If the neurons of a feature map did not share weights then the number of the trainable parameters of the convolutional layer would be the number of layer's neurons multiplied by the size of the kernel. In the example above it would be (217×217×56)×(5×5+1) or 68,561,584 parameters.

In addition to convolution, the values of a feature map are passed through a nonlinear function, such as a ReLU, to increase the non-linearity of the output and hence its accuracy.

### Pooling layer

CNNs often have pooling layers in order to reduce the dimensionality of the output of a convolutional layer (Goodfellow et al., 2016, chapter 9). The most popular type of pooling layer is called max pooling. It operates like a convolutional layer, meaning kernels of specified size are moved across the height and the width of an input image to generate new images (feature maps). However, max pooling returns the maximum value from the portion of the image covered by the kernel.

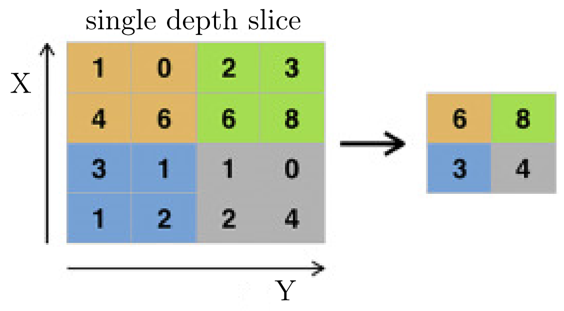


Figure 3.11: Max pooling process (Traore et al., 2018).

Since a pooling layer decreases the spatial dimensions of the preceding convolutional output, and thus the number of pixels in the feature maps, it speeds up the computational operations of the network. This characteristic of the network down-sampling the feature maps of the previous convolutional layer is referred to as spatial sub-sampling.

Another reason for the application of a pooling layer is the sensitivity the convolutional outputs have to the position of a feature in the input. This means that minor changes in the location of a feature in the input image will result in a different feature map. Small changes of a feature's position can occur through rotation or by cropping the input image. By down-sampling the feature maps, the objects detected in the input image will have the same location in the output even if they are slightly moved. This results from the pooling layer's capability of summarizing the presence of features in patches of the feature map. This characteristic of the pooling layer adds the model an amount of invariance to local translation and makes the extraction of features more robust.

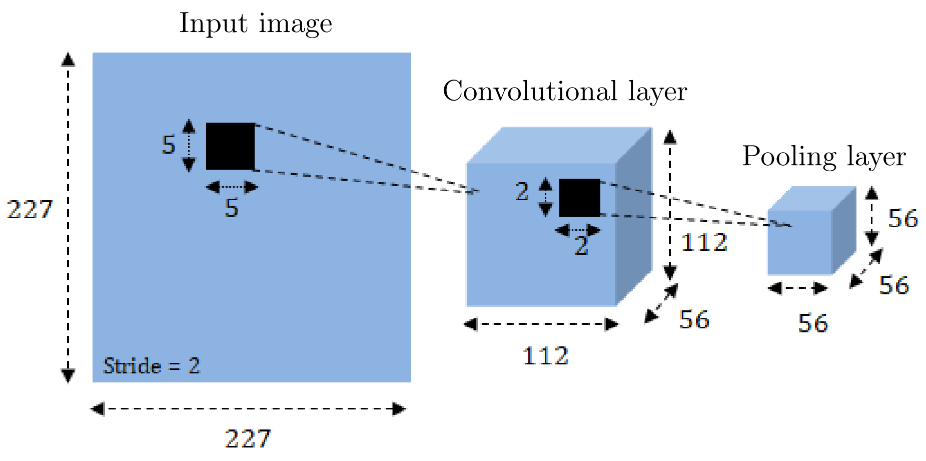


Figure 3.12: Illustration of a convolutional layer and a pooling layer.

Figure 3.12 is an extension of the previous one that includes a pooling layer. The kernel of this layer has a size 2×2 and moves with a stride of 2 across the height and the width dimension of the convolutional output. That results in a pooled output with half dimensions.

Compared to the convolutional layer, the pooling layer has no learnable parameters. There are no activations occurring in this layer, only spatial sub-sampling. Essentially, the pooling method is a fixed function meant to highlight the features in the previous output.

Stacking a pooling layer over a convolutional layer can be repeated one or more times when designing a CNN. This allows a hierarchical decomposition of the input. The first convolutional outputs will extract low-level features such as lines and edges of the input image. The next convolutional outputs will extract more complex features and the very deep layers will extract whole shapes, like faces or houses. The reason relies on the fact that the receptive field of the neurons of a deeper layer will encompass more information compared to the neurons of the previous layer because of the spatial down-sampling of the feature maps resulting in summarized versions of the input.

#### Fully-connected layer

After passing the input image through the convolutional layers, the model is enabled to understand the image features. For classification purposes, the final product of the convolutional layers has to be passed through a regular neural network to output predictions. In order for the output matrix to be passed as input to a regular hidden layer, it has to be flattened. That means, it is converted into a 1-dimensional array or vector. This vector is given as input to a fully-connected layer. The neurons of this layer are fully-connected to all the values of this vector.

The output can be computed by multiplying the vector values with the weights of each neuron adding the corresponding bias. Then, the weighted sum is passed through an activation function to increase the output accuracy. Usually, at the last fully-connected layer, the activation function that is applied is the softmax. Softmax function is similar to the sigmoid function, but it is used for multi-classification, whereas the sigmoid is used for binary classification (Nwankpa et al., 2018). The output of the model is a vector containing as many probabilities as the total number of class labels. The class label with the highest probability is the final class label.

## 3.5 Object detection algorithms

Apart from image classification, convolutional neural networks are widely used for object detection tasks. A CNN will take an image as input and divide it into various regions. Each region will be considered as a separate image and it will be passed to the CNN to be classified. Then, all the regions will be combined to form the original image that will now contain the classified objects. However, since the objects in an image can have different size and shapes, the CNN may need to divide the image into a massive number of regions, spending a huge amount of computational time. Research scientists optimized the object detection process and reduced the number of regions by creating a region-based CNN whose operation is summarized below.

### 3.5.1 Region-based Convolutional Neural Networks

The region-based algorithm, called R-CNN, involves three phases. In the first phase, it selects the regions of an image using a proposal method. It suggests a number of regions in the image based on a selective search algorithm and checks if any of these regions contain an object (Girshick et al., 2014).

The selective search algorithm divides an image into multiple regions and combines similar regions, based on size, color, and texture similarity, as well as shape compatibility, to form bigger regions (Uijlings et al., 2013). These regions are the region proposals, or as they called Regions of Interest (RoI), that the model will use for object detection, and are hand-labeled with a class and a ground-truth bounding box. A ground-truth bounding box is the (x,y) coordinates of an object in the image.

The second phase of the R-CNN involves the use of a deep convolutional neural network, often called backbone, as a feature extractor. The backbone network is pre-trained on a large auxiliary dataset. The last layer of the network is retrained based on the number of classes that need to be detected. Before entering the backbone network, the Regions of Interest are reshaped to match the required size of the network's input. Then, each of them is passed through the CNN for feature extraction.

The last phase of the R-CNN is classification. The extracted features and labeled class of each RoI are used to train SVMs (Support Vector Machines). A SVM is a machine learning algorithm that implements binary classification. For each class, one SVM is trained. Each SVM determines whether a RoI contains a specific class. Furthermore, a linear regression model, called bounding box regressor, is trained, by using the extracted features and the ground-truth bounding box of each RoI, to predict the bounding box for each identified object in the RoI. The architecture of the R-CNN model is summarized in Figure 3.13.

Unfortunately, the R-CNN model is slow in processing because it

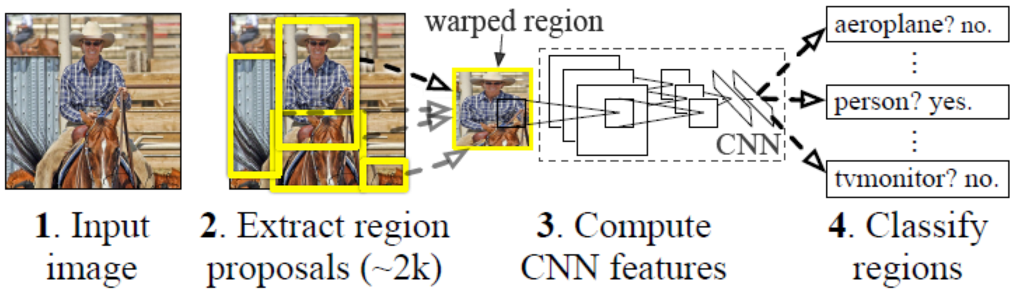


Figure 3.13: R-CNN workflow overview (Girshick et al., 2014).

divides an image into a large number of region proposals, hence it will need a lot of CNN forward propagations to perform object detection. Therefore, its architecture needed optimization.

Girshick Ross (2015) proposed an updated version of the R-CNN model and called it the Fast R-CNN. The main improvement in this model is that the pre-trained CNN feature extractor will be used in the entire input image. Then, the selective search algorithm will use the output feature maps to extract the region proposals. Each of these regions are inserted to a custom layer called region of interest pooling layer. This layer down-samples each region into a specified fixed-size feature map, regardless of the region's shape. Thus, it can extract features of the same shape even when region proposals have different shapes.

Since the output feature maps have the same shape, they can pass through fully-connected layers (FCs) which transform them into feature vectors. Each feature vector output bifurcates into two outputs. One output, produced by a softmax layer, is the class prediction. The other output, produced by a bounding box regressor, is the bounding box prediction. The workflow of the Fast R-CNN is depicted Figure 3.14.

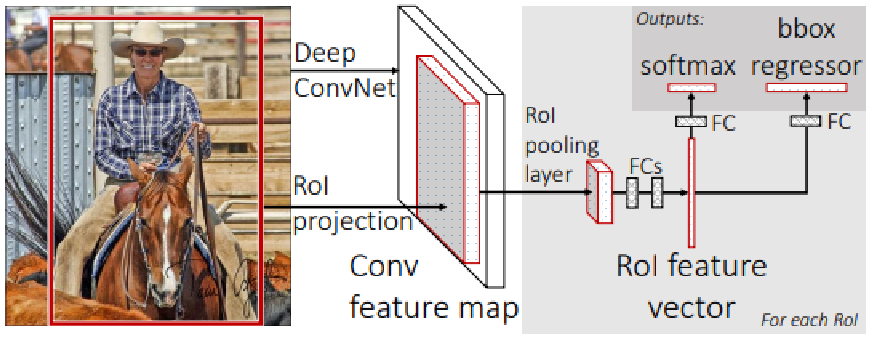


Figure 3.14: Fast R-CNN architecture summary (Girshick, 2015).

To increase accuracy in object detection, the Fast R-CNN model has to generate a lot of region proposals with the selective search algorithm. To maintain a good accuracy without generating too many regions, the model architecture had to be further improved.

The optimization of the model came with the replacement of the selective search algorithm by a Region Proposal Network (RPN). The RPN takes the output of the pre-trained convolutional neural network (backbone) and outputs a set of rectangular objects proposals. More specifically, the RPN slides a n×n spatial window over each output feature map and, at each sliding-window location, it generates k number of region proposals of different predefined shapes and sizes, called anchors. For each anchor, the RPN computes the probability that an anchor is an object, referred to as the objectness score, and predicts its bounding box (Ren et al., 2017).

The rest of the model remains the same. The anchors are passed through a RoI pooling layer to be transformed into the same size. Then, they enter a fully-connected network with a softmax layer and a bounding box regression layer to extract the class label and the bounding box of each object. Figure 3.15 depicts the model's workflow.

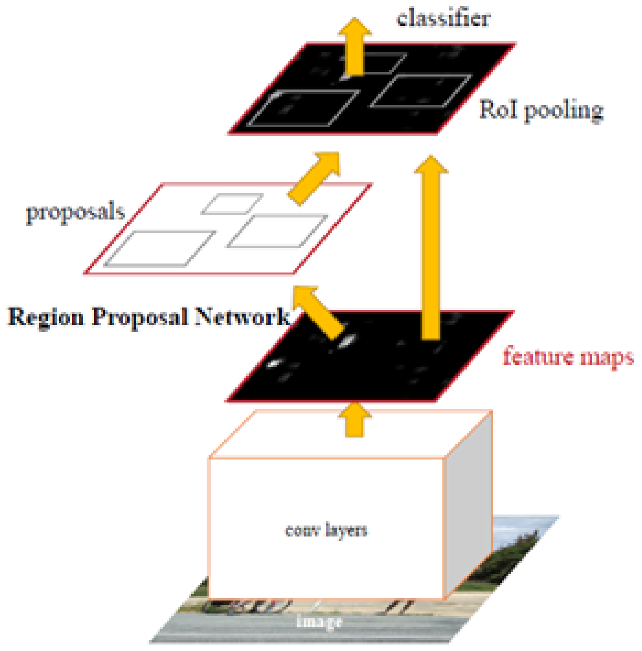


Figure 3.15: Faster R-CNN model architecture (Ren et al., 2017).

Faster R-CNN model architecture has the advantage that the RPN can be jointly trained with the rest of the model. Therefore, the parameters in the feature extractor CNN can be fine-tuned to:

- generate high-quality region proposals from the RPN, and
- maintain a good accuracy in the prediction of class and bounding box coordinates from the FCs,

at the same time.

The model's ability to recognize objects in an image was further extended to locate the exact pixels of an object instead of just its bounding box coordinates. Thus, the extended model would return three things for each object in an image:

1. its class label,
2. its bounding box coordinates, and

3. its segmentation mask.

The segmentation mask is a pixel-wise mask of an object that reveals the object's exact location in an image. Because of the additional feature of creating pixel-wise masks for the objects in an image, the model was called Mask R-CNN (He et al., 2017).

Compared to the structure of the Faster R-CNN model, in the structure of the Mask R-CNN the RoI pooling layer was replaced with a layer, called RoI alignment layer (RoIAlign). The reason behind this replacement is that the regions of a feature map selected by the RoI pooling layer were slightly misaligned from the regions of the original image. Although this small misalignment does not pose a problem when predicting bounding boxes, pixel-level segmentation requires pixel-to-pixel alignment between network inputs and outputs. The RoIAlign preserves the spatial information on the feature maps and achieves the desired alignment by using bilinear interpolation. Similar to the RoI pooling layer, the RoIAlign produces feature maps of the same shape for all region proposals.

The Mask R-CNN model architecture includes the integration of a fully convolutional neural network (Long et al., 2014) in the Faster R-CNN structure. This network takes as input a RoI feature map and outputs a binary mask of the feature map's object. The binary mask shows whether or not a given pixel of a region proposal is part of the object that it contains.

Figure 3.16 depicts the workflow of the Mask R-CNN model. Each anchor, produced by the RPN, is passed through the RoIAlign to be converted into a fixed-size feature map. It is then transferred to the FC layer branch for class and bounding box prediction and, in parallel, it gets into a fully convolutional network for binary mask prediction.

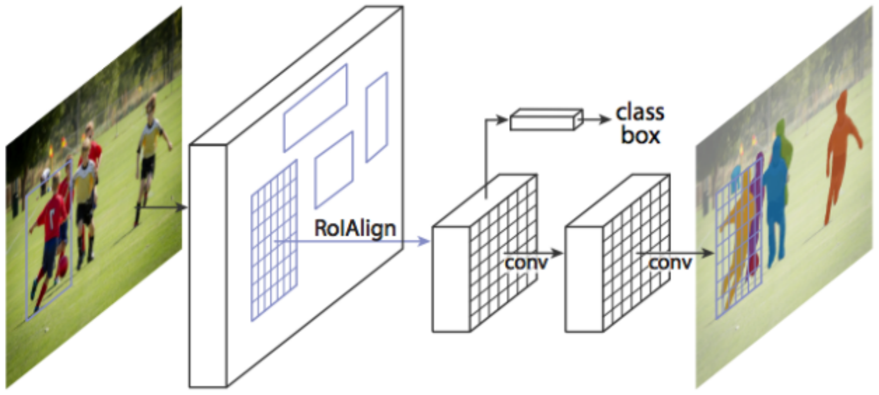


Figure 3.16: Mask R-CNN model's instance segmentation branch (He et al., 2017).

The Mask R-CNN model is developed to implement simultaneous detection and segmentation. This task is commonly referred to as *instance segmentation* - the recognition of the pixel-level regions of each object instance in an image. In the present study, instance segmentation is one of the tasks that have to be put into effect in order to achieve the objectives stated in the next section.

## 3.6 Evaluation metrics

After creating an algorithm capable of making predictions, one has to evaluate its predictive power. Whether the task is object detection or instance segmentation, there are commonly used statistics that indicate the effectiveness of an algorithm.

### 3.6.1 Precision, recall, and F1 score

Precision, also known as positive predictive value, measures the percentage of the predictions that are correct, i.e. the proportion of true positives:

$$Precision = \frac{TP}{TP + FP} \tag{3.9}$$

where $TP$ is the number of true positive predictions, and $FP$ is the number of false positive predictions (Ting, 2011).

Recall, also known as sensitivity, measures how well the model finds correct predictions, or in other words, finds the proportion of true positives out of all predictions that are and should be positive:

$$Recall = \frac{TP}{TP + FN} \tag{3.10}$$

where $FN$ is the number of false negative predictions (Ting, 2011).

Precision and recall values are between 0 and 1. When precision is high, the model is more confident when it classifies a sample as positive. When recall is high, the model is more confident that it classifies samples correctly as positive.

However, precision and recall cannot be both high because they are on opposite ends of a scale. When precision is high and recall is low, the model correctly gives a label to an object, but it fails to find many objects with the same label resulting in having many false negatives. When precision is low and recall is high, the model finds every true positive sample, but it also classifies many samples incorrectly as positive.

For every algorithm, it has to be decided whether it is more important to avoid false positives or false negatives. If it is more important to avoid false positives, the model is set to focus on having higher precision than recall. If it is more important for the model to predict correctly the true positives, it should achieve a higher recall. Deciding the best values for precision and recall can be a complex process. Researchers calculate a metric, called the F1 score, to find a balance between precision and recall where both of them will be as high as possible.

F1 score is defined as the harmonic mean of precision and recall and is estimated according to Equation 3.11:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{3.11}$$

A high F1 score indicates that both precision and recall are high, whereas a low F1 score signifies imbalance between the two metrics.

### 3.6.2 Intersection over union

Precision and recall is calculated to evaluate how often the algorithm predicted the bounding boxes and class labels correctly and if it predicted them correctly every time. To measure precision and recall for a bounding box prediction in terms of its class label is easy. To determine precision and recall based on whether the predicted bounding box surrounds sufficiently an object as the annotated one (ground truth) does, an additional metric is needed. This metric is the intersection over union (IoU) (Rezatofighi et al., 2019).

The intersection over union is a metric commonly used to evaluate any algorithm that outputs predicted bounding boxes. The IoU, also known as the Jaccard index, is a statistic used for comparing the similarity between two arbitrary shapes and is given by Equation 3.12:

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{3.12}$$

The IoU values are in the range of [0,1], the numerator denotes the area of overlap between shape A and shape B, and the denominator signifies the area of union between shape A and shape B. The IoU is very popular among object detection models because it is a scale invariant statistic. It encodes the shape properties of the predicted bounding boxes and the ground truth bounding boxes into the region property and then calculates a normalized measure focused on their areas (Rezatofighi et al., 2019).

If the IoU is above a certain threshold for a detection of a certain class, the detection is classified as true positive, otherwise it is classified as false positive. Each threshold may give different predictions from other thresholds. Usually, the predefined threshold is set to 0.5, but depending on the problem and dataset, a researcher might set a higher threshold that seems to be more reasonable.

For segmentation evaluation, the intersection over union, often called mask-to-mask IoU, measures the number of pixels common between the annotated masks (ground-truth) and the prediction masks divided by the total number of pixels present across both masks. The intersection over union for a segmentation task is illustrated in Figure 3.17.

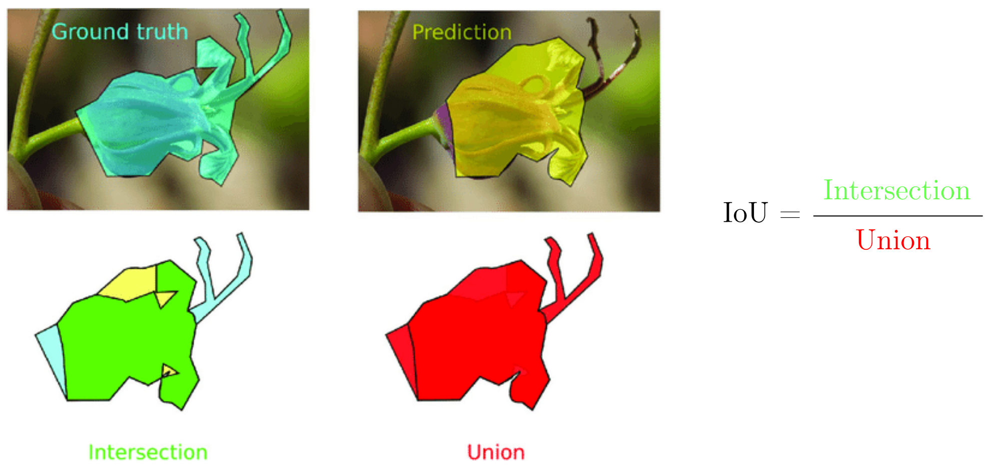


Figure 3.17: The ground-truth and the predicted segmentation mask of a flower used to compute the intersection over union metric (Goëau et al., 2020).

As with object detection IoU, a mask-to-mask IoU value is compared to an IoU threshold to determine whether a segmentation mask prediction is considered a true positive.

### 3.6.3 Confidence score

Object detection models make predictions in terms of bounding boxes and class labels. The output of these models include:

- class probabilities - conditional class probabilities of a detected object belonging to a particular class:

$$conditional\ class\ probability = P(class_i|object) \tag{3.13}$$

- bounding box coordinates - each bounding box include the (x,y) coordinates of its top left corner, its width and height dimensions, and a confidence score representing the probability of the box containing an object given by:

$$box\ confidence\ score = P(object) \times IoU \tag{3.14}$$

If there is not an object in the bounding box, the confidence score should be equal to zero. Otherwise, it should be equal to the intersection over union between the predicted box and the ground-truth.

At test time, the conditional class probabilities are multiplied with the box confidence score to acquire a class-specific confidence score (Redmon et al., 2015). This score measures the confidence on both classification and localization of the object in the image for each predicted bounding box. Thus, it is given by:

$$\begin{aligned} class\ confidence\ score &= P(class_i|object) \times P(object) \times IoU \\ &= P(class_i) \times IoU \end{aligned} \tag{3.15}$$

A prediction is considered a true positive (TP) if it satisfies two conditions:

- the predicted class matches the actual class (ground-truth label), and
- the IoU between the predicted bounding box and the ground-truth is greater than the IoU threshold.

When one of the two conditions is not met, the prediction is considered a false positive (FP). The PASCAL VOC detection challenge includes an additional condition to define true and false positives. For multiple detections of the same object, only the detection with the highest confidence score is counted as a true positive, while the rest of them are counted as false positives.

If the object detector fails to find an existing object in an image, it means that it made a false negative (FN) prediction. When computing the recall, there is no need to count the false negatives because the sum of true positives and false negatives equals the total ground-truths (the number of objects supposed to be detected). If the detector correctly predicts that an object does not exist in an image, it means that it made a true negative (TN) prediction. Nevertheless, in object detection tasks, evaluation metrics that include true negative detections are not considered since there is no point to count all possible bounding boxes that should not be detected in an image.

### 3.6.4 Average precision

An object detection algorithm can be evaluated using multiple IoU thresholds, instead of considering only one. For each threshold value, the detections are characterized as true positives and false positives and they are ordered based on their class confidence score in a descending order. Then, the precision-recall values are calculated based on the accumulated true positives and false positives values. These precision-recall values are used to create a curve known as the precision-recall curve.

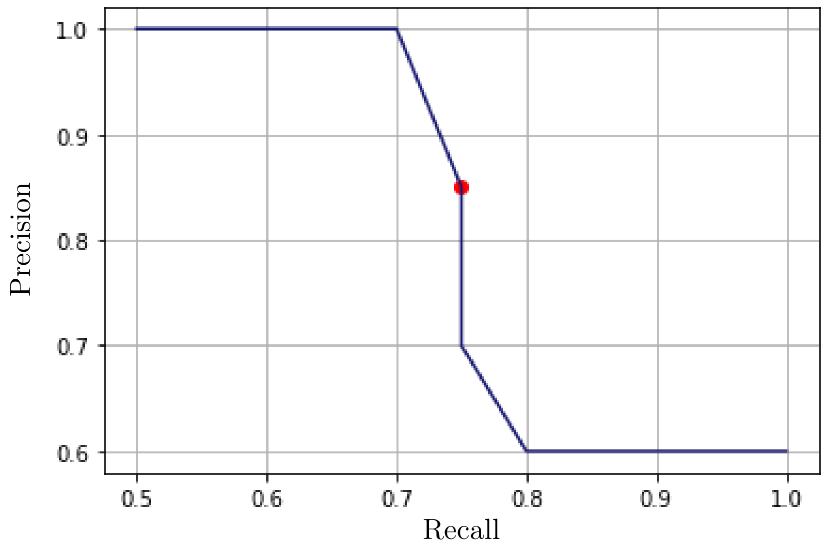


Figure 3.18: The precision-recall curve. The red dot represents the point where both precision and recall are high.

As shown in Figure 3.18, precision decreases when recall increases.

This happens because when the model predicts multiple positive samples, the probability of accumulating false positives increases leading to lower precision. The red dot represents the optimal precision-recall pair. This finding can be confirmed by calculating the F1 score for each precision-recall pair. The highest F1 score corresponds to the optimal precision-recall datapoint, where both precision and recall are high.

To summarize the precision-recall curve, researchers used a single metric that represents the average of all precisions. This metric is called the average precision (AP) and it is equal to the area under the precision-recall curve:

$$AP = \int_0^1 P(R)\,dR \tag{3.16}$$

The average precision values fall in the range of [0,1] because precision and recall values fall within this range. This metric is computed by drawing a precision-recall curve for one or multiple values of IoU threshold. The integral is approximated by the weighted sum of precisions, at each IoU threshold, where the increase in recall is used as the weight. It is given by Equation 3.17:

$$AP_{IoU} = \sum_n (R_{n+1} - R_n)\ P(R_{n+1}) \tag{3.17}$$

For each object class, the average precision, computed with Equation 3.17, is averaged over all IoU thresholds (if there are more than one):

$$mAP_{class} = \frac{1}{N} \cdot \sum_{i=0}^{N} AP_i \tag{3.18}$$

with $N$ being the total number of IoU thresholds. This is referred to as the mean average precision (mAP) of a specific class. To measure the accuracy of an object detection algorithm over all classes, researchers take the mean of the average precisions, as estimated by:

$$mAP = \frac{1}{N} \cdot \sum_{i=0}^{N} mAP_i \tag{3.19}$$

where $N$ is the total number of class labels.

The exact calculation of average precision varies between different detection challenges. For the 2007 PASCAL VOC object detection challenge, AP was computed through an 11-point interpolation, using one IoU threshold which was 0.5 (Padilla et al., 2020). The recall value for this IoU threshold was divided to 11 equally spaced recall levels (R) in the range of [0,1], and the average precision was the average of maximum precision value for these 11 points, as given by:

$$AP_{11} = \frac{1}{11} \cdot \sum_{R \in \{0.0, 0.1, ..1.0\}} P_{interp}(R) \tag{3.20}$$

where,

$$P_{interp}(R) = \max_{\tilde{R} \geq R} P(\tilde{R}) \tag{3.21}$$

The PASCAL VOC challenge has 20 different classes, and for each class the average precision was calculated using Equation 3.20. Then, the mean of these average precisions (mAP) was computed over the 20 classes. The reason behind this interpolation method was to reduce the impact of the "wiggles" in the precision-recall curve because the curve has often a zigzag-like pattern posing challenges to an accurate measurement of the AP. Therefore, researchers used this method in order for the precision to decrease monotonically. However, the method of 11-point interpolation using was not very precise and it could not measure differences with models having low average precision.

Later visual recognition competitions, such as PASCAL VOC 2010-2012 and Open Images challenge (Google Research, 2019), measure the average precision as the area under the precision-recall curve for an IoU threshold of 0.5. The method of estimating the average precision is referred to as the all-point interpolation and is given by:

$$AP_{all} = \sum_{n} (R_{n+1} - R_n)\ P_{interp}(R_{n+1}) \tag{3.22}$$

where,

$$P_{interp}(R_{n+1}) = \max_{\tilde{R} \geq R_{n+1}} P(\tilde{R}) \tag{3.23}$$

This method samples precision at every recall level, taking the maximum precision whose recall is greater or equal than $R_{n+1}$.

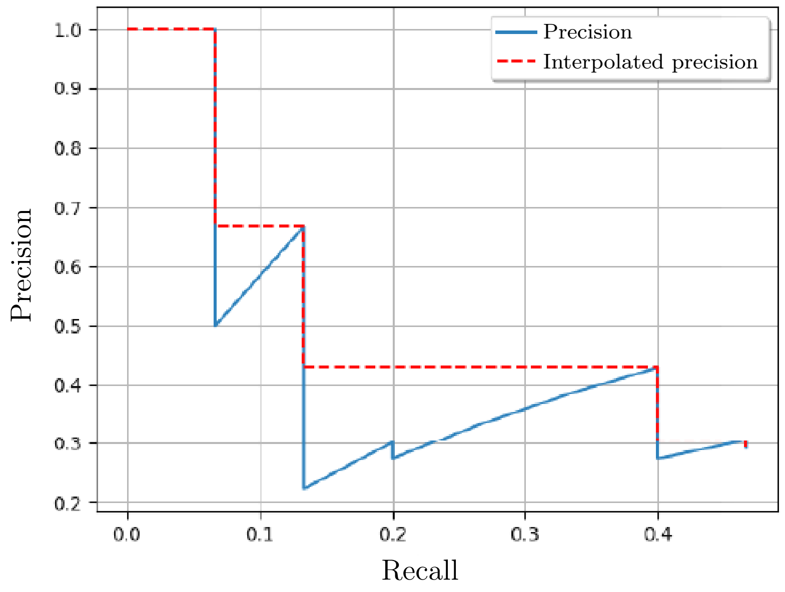


Figure 3.19: Difference between the actual precision-recall curve and the interpolated curve (Padilla et al., 2020).

As depicted in Figure 3.19, all-point interpolation method approximates the area under curve and computes the average precision as the sum of the rectangular blocks.

In PASCAL VOC competition, if there are five detections of a single object, only one with the highest IoU is counted as correct detection and the rest four are false detections. In Google's Open Images competition, the object classes are organized in a semantic hierarchy, meaning an object can belong to a general category and a subcategory. Thus, both the category and the subcategory of a given detection should be reported. If only the subcategory label is correctly produced for an object detection, the detection is counted as true positive but the undetected category label will be counted as a false negative.

In Google's Open Images challenge, the mean average precision is estimated, not only for the object detection task, but also, for instance segmentation using a mask-to-mask IoU threshold of 0.5 across the 300 classes of its dataset. As in the object detection case, the mean average precision needs to handle the semantic hierarchy of the object classes.

The Microsoft's COCO challenge (Lin et al., 2015) uses an 101-point interpolated precision to estimate average precision for object detection and instance segmentation. The precision is interpolated at 101 recall levels (i.e., 0.0, 0.01, 0.02, ....1.0) and the average precision is computed for:

- a single IoU threshold of 0.5,
- a single IoU threshold of 0.75, and
- multiple thresholds starting with a minimum of 0.5 and reaching the maximum of 0.95 with a step size of 0.05. This method of computing average precision is denoted as AP@[.50:.05:.95].

Average precision for 10 IoU thresholds is calculated because it rewards detectors with better localization. The final average precision is calculated over 80 categories, but it is still referred to as average precision (AP) instead of mean average precision (mAP).

### 3.6.5 Average recall

For the COCO challenge, except for the average precision, there are other evaluation metrics for the object detection or segmentation task. One of them is the average recall (AR), which summarizes the distribution of recall across a range of IoU thresholds. According to Hosang et al. (2016) this metric is equal to twice the area under the recall-IoU curve between 0.5 and 1, given by:

$$AR = 2 \cdot \int_{0.5}^{1} R(IoU)\, dIoU \tag{3.24}$$

The average recall estimated for the COCO dataset, corresponds to the maximum recall given a fixed number of detections per image (1, 10, or 100) averaged over all IoUs and over all classes similarly to average precision (Padilla et al., 2020). The average recall for 1 detection per image is denoted as $AR^{max=1}$, for 10 detections per image as $AR^{max=10}$, and for 100 detections per image as $AR^{max=100}$.

### 3.6.6 Average precision and recall across scales

Additional metrics have been defined for the COCO challenge, like the average precision across scales and the average recall across scales. These metrics are used for evaluating detections based on the object size. More specifically, they are determined for:

- small objects with an area less than $32^2$ pixels,
- medium objects with an area between $32^2$ and $96^2$ pixels, and
- large objects with an area greater than $96^2$ pixels.

The calculation of these statistics will give an idea of the model's ability to detect well either small or large object or objects of varying sizes.

### 3.6.7 Average segmentation accuracy

The PASCAL VOC 2012 challenge judges segmentation tasks by average segmentation accuracy computed across the twenty classes and the background class. Segmentation accuracy is the per-pixel accuracy calculated for a certain mask-to-mask IoU threshold. It is defined as the percentage of correctly predicted pixels out of all predictions, and it is calculated for each class as follows:

$$segmentation\ accuracy = \frac{TP}{TP + FP + FN} \tag{3.25}$$

Then, this metric is averaged across the classes (including the background class) to obtain the overall segmentation accuracy.

When calculating an accuracy metric to evaluate a segmentation or object detection task, it should be noted that the class distribution has to be uniform in a dataset. The number of instances for each class has to be the same in all the other classes. Otherwise, this statistic can be very misleading because it will not represent equally the performance of the model across all classes.

# Chapter 4

# Methodology

## 4.1 Introduction

The experimental work involved data collection and pre-processing, data inspection, object detection algorithm training and inference, and evaluation of the results. The study goal was to experiment on litter detection using a deep learning object detection algorithm trained on aerial image data, and on regular image data collected in the field. Processing of aerial imagery was selected because it provides smaller spatial resolution than regular images. Thus, the task of the algorithm was to detect relatively small sized litter objects in the images compared to regular litter image data. Another reason of using aerial imagery is that there is a lot of recent research, mentioned on the preceding chapter, that processes images obtained from a UAV for litter detection. The regular image dataset was used for training an object detection algorithm to localize and classify litter by type and by material - something that is difficult to happen on aerial imagery due to the small size of objects. Before discussing data collection and data processing configurations, this chapter begins with a detailed analysis of the algorithm architecture applied to the data.

## 4.2 Algorithm architecture

In addition to locating and classifying litter in image data, this research also aimed to perform instance segmentation. In other words, it aimed to differentiate objects in image data by providing masks in pixel level of each litter item in an image along with an object bounding box and a class label. For that reason, the algorithm selected to solve this task was the Mask R-CNN (He et al., 2017). This algorithm structure and internal workings were briefly explained in chapter 3, but here follows an even more detailed explanation.

### 4.2.1 Feature Pyramid Network

The Mask R-CNN algorithm is characterized as a two-stage object detector and both stages are connected to the backbone structure depicted in Figure 4.1. The backbone structure, called the Feature Pyramid Network (FPN), is proposed by Lin et al. (2017) as a high efficient architecture for the object detection task. It involves a bottom-up pathway, a top-down pathway and lateral connections.

The bottom-up pathway is the backbone convolutional neural network that extracts the features from the input image. The backbone neural network used in this project is a Residual Network with 101 layers (ResNet101), introduced by He et al. (2016). It outputs feature maps at several scales with a scaling step of 2. This results in having levels of feature maps in a pyramidal shape as shown in Figure 4.1. Each pyramid level is represented by the last feature map of each of the network's stages. ResNet101 architecture has 4 stages of convolutional layers, therefore the output of the last layer of each stage is used as the reference set of feature maps. The reference set of feature maps is called {C2, C3, C4, C5}, named from the convolutional layers in ResNet architecture, and it is used for the creation of the feature maps, in the top-down pathway, of the same spatial size at each pyramid level. In the top-down pathway, proportionally sized feature maps are generated from

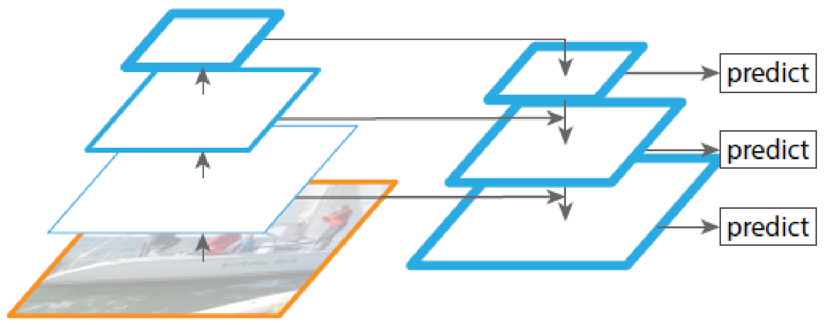


Figure 4.1: Feature Pyramid Network. The feature maps are indicated by blue outlines and thicker outlines denote semantically stronger features (Lin et al., 2017).

the higher pyramid levels, by up-sampling the spatial resolution with a factor of 2 using nearest neighbor up-sampling. These feature maps are enriched with the features from the bottom-up pathway via the lateral connections. Lateral connections represent a simple 1×1 convolution operation that reduces the dimensionality of the feature maps (the channel or depth dimension) without applying any spatial transformation, and thus retaining important feature-related information. This is because each layer in the bottom-up pathway has different number of channels and in order to use shared classifiers and bounding box regressors, the channel dimension needs to be fixed. As in the paper, the channel dimension is fixed with a value of 256. Because of the up-sampling, the dimensions of the feature maps at each pyramid level with their corresponding feature maps from the bottom-up pathway are the same and they can be merged by element-wise addition. Finally, a 3×3 convolution is applied to all merged feature maps to reduce the aliasing effect of up-sampling without affecting the spatial size of the feature maps. This final set of feature maps corresponding to {C2, C3, C4, C5} is called {P2, P3, P4, P5}. The full FPN architecture is shown in Figure 4.2.

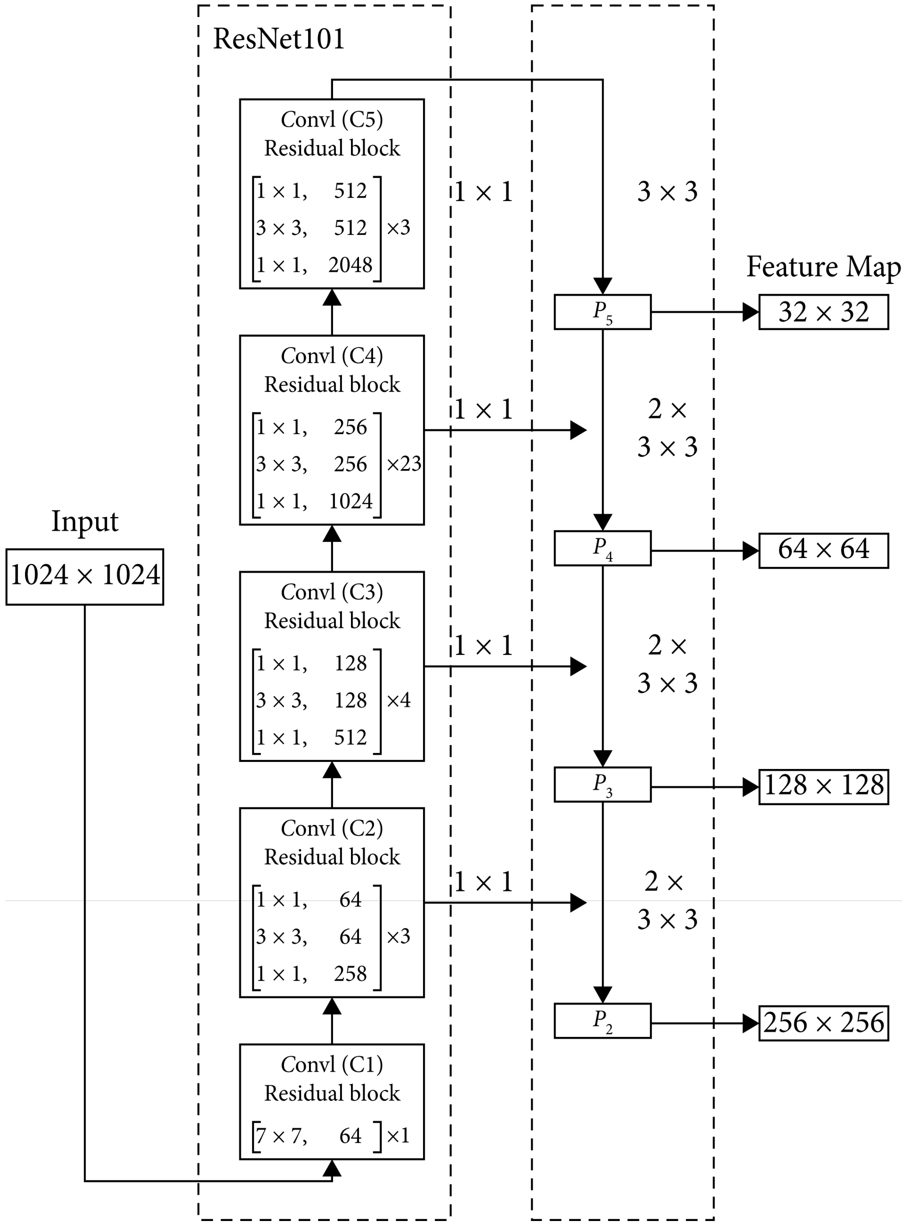


Figure 4.2: Full scheme of Feature Pyramid Network using ResNet101 (Zhang et al., 2021).

The importance of the FPN as a feature extractor structure comes from the enrichment of {P2, P3, P4, P5} set of feature maps from {C2, C3, C4, C6} set. This enrichment occurs because the features of the bottom-up pathway are more accurately localized as they have undergone sub-sampling fewer times. The features of the bottom-up are weaker semantically, but if they are combined with the features of the top-bottom pathway, which are of higher-level semantics, the network will provide strong semantically features at various resolution scales. This combination enhances the detection performance, especially for small objects.

### 4.2.2 Region Proposal Network

As it was mentioned in chapter 3, Mask R-CNN generates proposals about the regions where there might be an object based on the input image. It does so by using a light weight neural network called RPN (Region Proposal Network) which slides a 3×3 convolution kernel on each feature map of the top-down FPN pathway. At each scanning position, the RPN generates a set of boxes with predefined aspect ratios and scales relative to images, called anchors, to figure out an object's location in a feature map and its bounding box size. Anchors of a single scale and of multiple aspect ratios are assigned to each top-down pathway level, but the scale is different on each level. Specifically, the anchor scales are 5 of size $\{32^2, 64^2, 128^2, 256^2, 512^2\}$ for {P2, P3, P4, P5, P6} respectively. P6 is created so that the 5th anchor can be used. It is generated by sub-sampling P5 with a stride of 2. The aspect ratios of the anchor boxes are {1:1, 1:2, 2:1}, meaning that for each pixel in the feature maps, 3 anchor boxes are generated in one of the scales specified above. If the anchor boxes with the specified aspect ratios and 5 different scales were drawn on the center pixel of a 1024×1024 sample image, the result could be visualized in Figure 4.3.

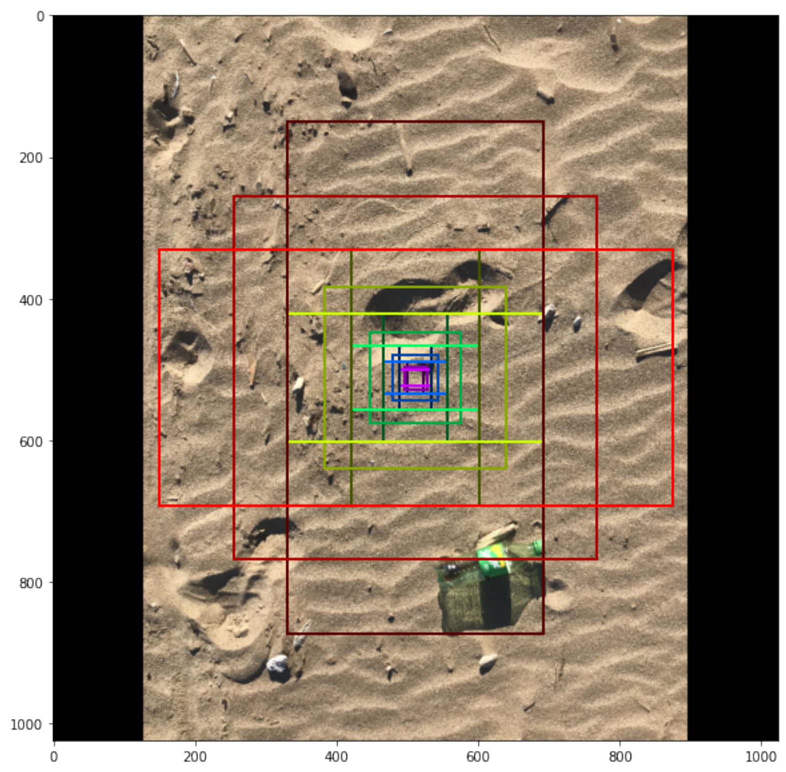


Figure 4.3: Anchor boxes at the center cell of a sample image. Each color represents a different anchor scale and for each color there are anchor boxes of 3 aspect ratios {1:1, 1:2, 2:1}.

The 3×3 convolution kernel that RPN uses to generate anchor boxes on {P2, P3, P4, P5, P6} is followed by two parallel 1×1 convolutional layers. RPN uses one of the two layers for binary classification, and it outputs the probability that an anchor is an object, referred to as the "objectness score". The other layer is used for bounding box regression, and it outputs 4 predictions, commonly called the deltas, $\Delta x$, $\Delta y$, $\Delta width$ , $\Delta height$, which are the offsets of an anchor box with respect to an object's ground-truth box. These deltas are applied to the anchors to generate the region proposals.

RPN's learning process results from the comparison of anchors boxes with the ground truth boxes using intersection over union (IoU) metric (section 3.6.2). If an anchor box overlaps a ground truth box with an

IoU over 0.7, it is classified as positive - it contains an object. If the IoU is below 0.3, it is classified as negative - it is background. If there is no anchor box with an IoU overlap higher than 0.7 but is over 0.5, then a positive label is assigned to the anchor(s) with the highest IoU overlap with a ground-truth box. The anchor boxes that do not match the conditions above, they do not contribute to the training (Ren et al., 2017). Figure 4.4 shows ten random anchor boxes generated for a sample image, from which one is positive and the other nine negative.

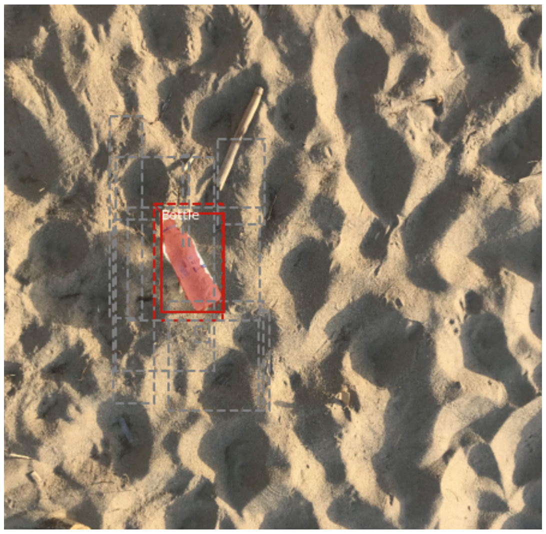


Figure 4.4: Random anchor boxes generated for a sample image. The solid red line represents the ground-truth bounding box, the dashed red line represents a positive anchor box, and the dashed grey lines represent negative anchor boxes.

Anchors are randomly sampled to form a mini batch of size 256, with a balanced ratio between positive and negative anchors (1:3) (Ren et al., 2017). RPN uses this mini batch to calculate the classification loss using the binary cross-entropy loss function given by:

$$L_{cls}(p_i, p_i^*) = \begin{cases} -\log(p_i), & p_i^* = 1 \\ -\log(1-p_i), & \text{otherwise} \end{cases} \tag{4.1}$$

where $p_i$ is the predicted probability of a sample being an object (corresponds to the ground-truth label $p_i^* = 1$), and $1 - p_i$ is the predicted probability of a sample not being an object. Then, it uses only the positive anchors of the mini batch to estimate the regression loss. For each of these positive anchors, the closest ground truth bounding box is used to calculate the coordinate offsets needed to transform the anchor into the object. The parameterization of the 4 offsets are:

$$\begin{aligned} t_x &= \frac{(x - x_a)}{w_a}, \ \ t_y = \frac{(y - y_a)}{w_a} \\ t_w &= \log(w/w_a), \ \ t_h = \log(h/h_a) \\ t_x^* &= \frac{(x^* - x_a)}{w_a}, \ \ t_y^* = \frac{(y^* - y_a)}{w_a} \\ t_w^* &= \log(w^*/w_a), \ \ t_h^* = \log(h^*/h_a) \end{aligned} \tag{4.2}$$

where, x, y, w, and h denote the predicted box's top left corner coordinates and its width and height (Ren et al., 2017). Variables $x_a$, $y_a$, $w_a$, $h_a$, denote the anchor box's coordinates, and $x^*$, $y^*$, $w^*$, $h^*$ denote the coordinates of the ground-truth box respectively. These offsets are given as input vectors to estimate the regression error using the smooth L1 function given by:

$$L_{reg}(t_i, t_i^*) = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t_i - t_i^*) \tag{4.3}$$

in which,

$$smooth_{L_1}(t_i - t_i^*) = \begin{cases} 0.5 \cdot (t_i - t_i^*)^2, & |t_i - t_i^*| = 1 \\ |t_i - t_i^*| - 0.5, & \text{otherwise} \end{cases} \tag{4.4}$$

It is likely that some anchors highly overlap with each other over the same object. To solve this issue and get the final region proposals, a simple algorithmic approach is adopted, called Non-Maximum Suppression (NMS). NMS takes as input the proposals sorted by their objectness score and discards those that have an IoU larger than some predefined IoU threshold, here 0.7, with a proposal that has a higher score (Ren et al., 2017).

The hyperparameters used in the RPN configuration are summarized in Table 4.1.

| Hyperparameters | Values |
|---|---|
| RPN anchor ratios | [0.5, 1, 2] |
| RPN anchor scales | (32, 64, 128, 256, 512) |
| RPN anchor stride | 1 |
| RPN train anchors per image | 256 |
| RoIs positive ratio | 0.33 |
| RPN NMS threshold | 0.7 |

Table 4.1: Hyperparameters applied to the Region Proposal Network.

### 4.2.3 Region of interest pooling

Region proposals, or region of interests (RoIs), are proposed bounding boxes where their coordinates are presented based on the original image size. In order to propagate them forward through the second stage of the Mask R-CNN network (multi-class classification, bounding box refinement, and mask generation branches), they should be mapped first to the feature maps of the pyramid levels, based on their size, and then, extracted from the different levels for predictions. A region proposal of width $w$ and height $h$ is assigned to the level Pk of the feature pyramid estimated using the formula below:

$$k = k_0 + \log_2(\sqrt{w \cdot h}/224) \tag{4.5}$$

The number 224 corresponds to the ImageNet competition pre-training size, and $k_0$ is the target level on which a region proposal with $w \times h = 224^2$ should be mapped into. Here, $k_0$ is equal to 4. After that, the proposals are rescaled to their corresponding feature map size, by dividing the coordinates with the scaling factor of the feature maps (Lin et al., 2017). For instance, if a feature map was decreased $n$ times from the original image, each coordinate of the proposal is divided by $n$. However, some coordinates divided by $n$ are float values, meaning the region proposal is not aligned with the grid of the feature map, as shown in Figure 4.5. This means, that the region proposal cannot be extracted from the feature map without losing some information. Furthermore, at the second stage of the Mask R-CNN network, the fully-connected layers used for object detection require fixed shape inputs to classify them into a fixed number of classes.

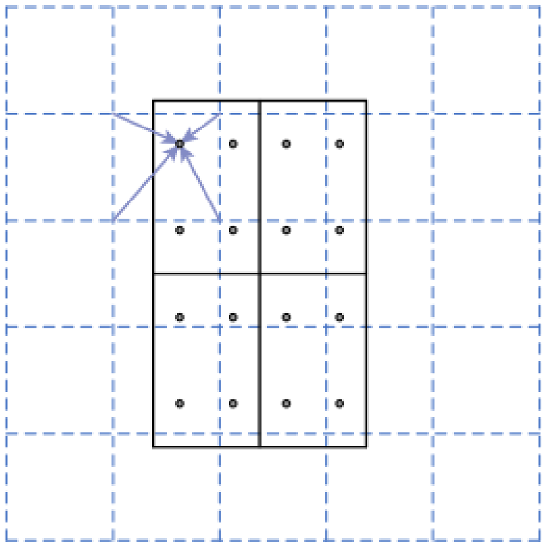


Figure 4.5: RoIAlign operation example that extracts a 2×2 region of interest. The dashed grid represents a feature map, the solid grid a region, and the dots the 4 sampling points in each bin. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map (He et al., 2017).

To compute the exact values of the region proposals in their respective feature maps and convert them into a fixed size to be used by the rest of the network, the proposals are passed through a special layer, called RoIAlign, which corrects the misalignment and extracts them as fixed size 7×7×256 feature maps. The region proposals are divided into 7×7 bins. Formally, in each bin 4 points are sampled using bilinear interpolation, and from these 4 points the maximum or average value is taken (He et al., 2017). Here, the interpolation comes from one sampling point from the center of each bin, which is nearly as effective.

### 4.2.4 Classification and localization

The fixed size region proposals are passed through two hidden fully-connected layers of size 1024 with ReLU activation. Then, for the final multi-classification they are passed through a fully-connected layer with $N + 1$ neurons, where $N$ represents the number of classes plus one for the background class. The activation function applied for the multi-classification is the softmax. In parallel, with the classification layer, there is another fully-connected layer that refines the proposals and generates the final bounding boxes. This layer has $4N$ neurons, because of the 4 offset predictions per class.

These fully-connected layers are trained in a similar way as RPN, by matching region proposals bounding boxes with ground-truth boxes using the intersection over union metric, but taking into account the different possible classes. The proposals with an IoU of at least 0.5 with a ground truth box are classified as positive, and are assigned to the class of the ground truth box. The ones with an IoU under 0.5 but over 0.1 with a ground-truth box are classified as negative/background. However, the proposals that do not intersect with a ground-truth box are ignored, because at this stage it is assumed that the proposals are good enough.

For detection, the region proposals are randomly sampled to form a mini batch of size 200 with a balanced ratio between positive and negative proposals of 1:3. The selected proposals are used to calculate the classification error by computing the binary cross-entropy for each class and then sum them:

$$L_{cls\_total} = \sum_{i} L_{cls}(p_i, p_i^*) \tag{4.6}$$

The proposals labeled with the correct class, and matched to a ground-truth box are used as input to the smooth L1 loss function defined by Equation 4.3 to estimate the regression loss.

The class predictions are in the form of a vector of class probabilities. Each proposal is labeled with the class having the highest probability. The proposals that have the background class as the highest probability are ignored. Then, the offset predictions are applied to the proposal to adjust the bounding box. But in order to get the final results, the proposals are grouped by class and sorted by probability values. The ones having probability value less than a probability threshold are filtered out. The probability threshold is a hyperparameter and represents the minimum detection confidence value, here 0.9. Then, the filtered proposals are passed through Non-Maximum Suppression algorithm (NMS). For each class group, NMS discards the proposals that have an IoU larger than 0.3 with a proposal that has a higher probability. Also, the maximum instances output for each class is set to 100. Subsequently, the groups are joined again to form the final list of detected proposals.

### 4.2.5 Instance segmentation

In parallel to the fully-connected layers used for object detection, there is another small neural network responsible for instance segmentation, the task of generating masks for each object in the feature maps. Instance segmentation is a pixel-to-pixel task and it requires much more fine-grained alignment of the region proposals than bounding boxes. RoIAlign operation is used mostly to properly align the regions of interest to the feature maps of the FPN to benefit mask generation rather than object detection. For instance segmentation, RoIAlign layer extracts feature maps of shape 14×14×256. These maps are passed through a fully convolutional network (FCN), instead of fully-connected layers (Long et al., 2014). Fully-connected layers require fixed-size input vectors which means the spatial coordinates are discarded to produce non-spatial outputs like vectors of class labels and box offsets. However, instance segmentation is a dense prediction task because a label for each pixel is predicted, hence the spatial layout of the objects has to be preserved. The trick is to use fully-connected layers as convolutional layers with kernels that cover the entire input regions. For instance, a convolutional layer with a kernel of size 3×3 can slide through the width and height of the input with a stride of 1 and same padding, and according to Equation 3.8, the spatial dimensions of the input will not change. FCN consist of a stack of 4 convolutional layers with ReLU activation that preserve the spatial coordinates of the input, a transposed convolutional layer that up-samples the feature maps by a factor of 2, and a convolutional layer with a 1×1 kernel followed by a sigmoid activation function that produces a classification map for each region proposal.

The transposed convolutional layer uses a kernel of learnable parameters, here of size 2×2, and multiplies all the kernel values with the first value of the input matrix, and place this 2×2 block of pixel values in place of the initial pixel value. Then it does the multiplication with the second pixel and places the result in the corresponding position. It repeats the process with the next pixel value until the output matrix is filled. The specified stride by which the kernel moves across the input is 2. Long et al. (2014) state in their research they applied this method of up-sampling in FCN because it is fast and effective for learning dense prediction.

The last convolutional layer reduces the dimensionality of the input maps to the number of the classes the dataset has. If the number of classes is $K$, and the resolution of the region proposals is 28×28, the output of the last convolutional layer is $28{\times}28{\times}K$ for each proposal. Since for each proposal, one mask is generated for each of the $K$ classes, there is not competition among classes when the network is learning. The activation function applied to each mask is a per-pixel sigmoid, because the output is binary and the network has to predict for each pixel value whether it is 0 or 1.

To determine the mask loss, only the positive region proposals are taken into account. As stated in the previous section, the positive region proposals are the ones having an IoU of at least 0.5 and labeled with the ground-truth class $k$. Also, only the mask output of the proposal associated with the $k$ class contribute to the mask loss. The mask loss is computed using the average binary cross-entropy function given by:

$$L_{mask} = -\frac{1}{m^2} \sum_{1 \leq i,j \leq j} [y_{ij} \log \hat{y}^k_{ij} + (1 - y_{ij}) \log(1 - \hat{y}^k_{ij})] \tag{4.7}$$

where $y_{ij}$ is the class label of a pixel $(i, j)$ in the true mask of the proposal of size m×m, $\hat{y}^k_{ij}$ is the predicted label of the pixel $(i, j)$ in the predicted mask for the ground-truth class $k$.

The hyperparameters applied to the RoIAlign, detection and segmentation layers in Mask R-CNN are summarized in Table 4.2.

| Hyperparameters | Values |
|---|---|
| RoIAlign pool size | 7 |
| Training RoIs per image | 200 |
| Detection min confidence | 0.9 |
| Detection max instances | 100 |
| Detection NMS threshold | 0.3 |
| RoIAlign mask pool size | 14 |
| Mask shape | 28x28 |

Table 4.2: Hyperparameter configuration of RoIAlign, detection and segmentation operations.

## 4.3 Datasets and annotations

### 4.3.1 Aerial images

To implement the deep learning task of litter detection in aerial imagery, a field-specific training dataset was needed. For this purpose, a publicly available dataset consisted of low altitude drone imagery, namely the UAVVaste dataset (Kraft et al., 2021), was selected. The UAVVaste dataset consists to date of 772 images and 3716 annotations. Kraft et al. (2021) tested real-time algorithms on different kinds of processing hardware to propose the most efficient combinations for real-time litter detection. Here, it was attempted to use an object detection algorithm that has not be used before in the UAVVaste dataset.

The annotations in UAVVaste are instance segmentation annotations in the COCO format (Lin et al., 2015). That means they involve bitmaps containing a mask marking which pixels in the image contain each object. All objects in this dataset belong to one category - they are recognized as rubbish. The dataset was split into training, validation, and test set. Around 10% of the images were used for fine-tuning the object detection model after each epoch, and 1% were used for testing the model on unseen data for an unbiased evaluation.

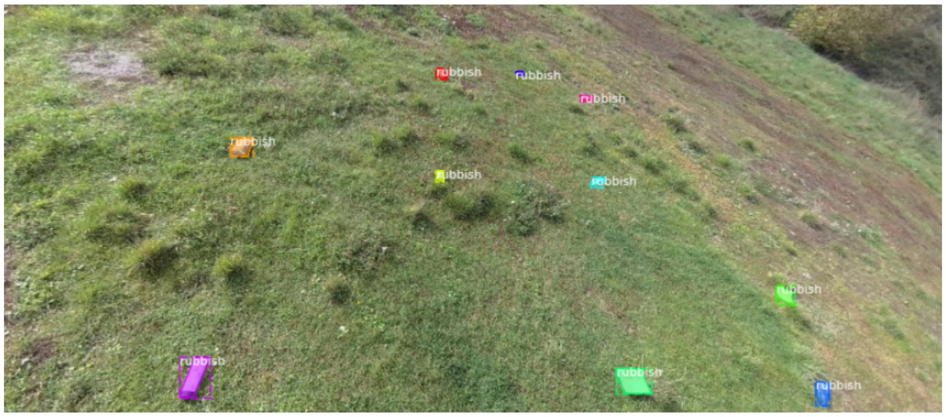


Figure 4.6: Annotated instances of UAVVaste dataset.

### 4.3.2 Coastline litter images

Apart from the UAVVaste dataset, Mask R-CNN algorithm was trained and tested on another dataset. Ground assessments were carried out in the coastal areas, around the city of Chania (Crete, Greece), to identify areas affected with litter pollution suitable for data collection. West of the city of Chania, the coastal area that starts from the Nea Chora beach and ends at the Golden beach, was selected because of its considerable transport of litter items from touristic activities.

The collected dataset, named CoastLitter, is comprised of 1635 annotated images and 2451 annotations. The annotations take the format of JSON files. This file format provides an easy way to store and share information in a pair attribute-value format. Images were uploaded to VGG Image Annotator tool (Dutta and Zisserman, 2019), an online image annotation tool which is a typical JSON file format used by Mask R-CNN. For each object a segmentation mask was drawn over it and

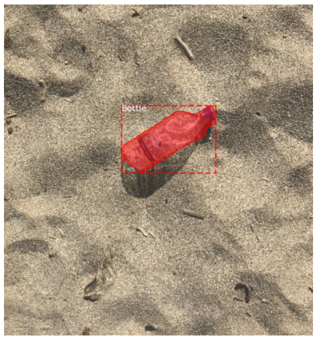


Figure 4.7: Annotated sample.

labeled as one of the 4 classes: bottle, cup, can, and other. Initially, each litter item was tagged by type and material (e.g., plastic cup, paper cup). However, because the dataset is small and there were not enough instances for each class, some classes were merged so the objects were classified by type, and separately, by material. The material selected to classify litter was plastic, hence the data were also classified as either plastic or non-plastic. In addition, for comparison purposes only with the UAVVaste dataset, another annotated dataset was formed, where all classes were merged into one class: the litter class. Around 15% of the annotated images were used for tuning the object detection model's parameters, and 8% for assessing its performance.

## 4.4 Training and configurations

The training of the Mask R-CNN algorithm was implemented on a Google Colaboratory notebook rather than on local machine, because Google Colaboratory (free service) runs in Google server and gives access to free GPU (Nvidia Tesla K80 GPU) and 12GB RAM for faster processing.

Since Mask R-CNN is a sophisticated algorithm to implement, instead of developing it from scratch, the reliable and wildly-used third-party implementation by Matterport Inc. was used. The source code, authored by Abdulla (2017) is open-sourced and available in a Github repository. The Matterport implementation of Mask R-CNN was written in Python and built with the TensorFlow-Keras deep learning framework.

To mitigate the problem of overfitting, transfer learning was applied by using as starting parameters the pre-trained weights of Microsoft's COCO dataset. All images were resized to a constant square size of 1024×1024. To save memory space, and thus improve training speed, instance binary masks were optimized by storing the mask pixels that are inside an object's bounding box, instead of the mask pixels of the full image. Also, the masks were resized to a smaller resolution of size 56×56, instead of 1024×1024. The effect of mask resizing can be visualized in Figure 4.8.

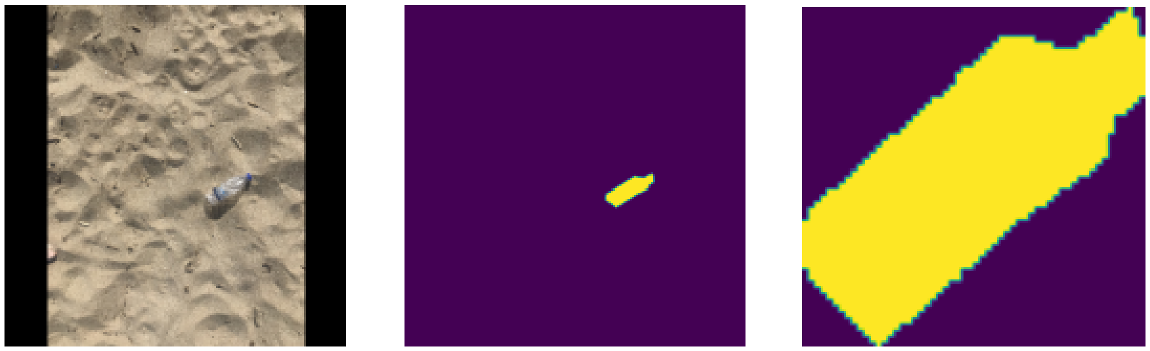


Figure 4.8: Mask resizing of a sample image.

An image augmentation sequence was created to be applied in the training phase of the algorithm. It applied affine transformations to images like changes to scale, translations and rotations, horizontal flips, and also it added a bit of Gaussian noise and blur as well as changes to brightness.

Mask R-CNN was trained in two stages. The first stage involved the training of only the randomly initialized layers, which are those layers in which no pre-trained weights have been used, and all the backbone layers were frozen. The second stage involved the fine-tuning of all layers. The learning rate applied in the Mask R-CNN paper implementation (He et al., 2017) was 0.01, but for this project it was too high, and therefore, it was reduced to 0.007. The batch size used in this implementation of Mask R-CNN algorithm was set to 1 as it worked well with the small learning rate, accelerating the learning process. On the other hand, a large batch size provided better gradient estimates, but this came at a cost of computational efficiency and good generalization performance. The optimization algorithm used to minimize the loss function was the stochastic gradient descent (SGD) (Ruder, 2017). It was stated in section 3.3.1. that gradient descent updates the weights and bias until it reaches the local minimum of the loss function using Equation 3.3. To accelerate the convergence speed, an extension to the gradient descent algorithm was used, referred to as momentum (Ruder, 2017). The momentum acts as an additional hyperparameter to Equation 3.3 where a fraction of history (gradient of the previous timestep) is added to the current gradient. With this additional hyperparameter, the update of parameters is influenced by the previous change. As shown in Equation 3.3, the change in training parameters is calculated as the gradient of the parameter $\theta$ in the current iteration scaled by the step size or learning rate $\alpha$ :

$$\Delta\theta = \alpha \cdot \nabla f(\theta) \tag{4.8}$$

Gradient descent with momentum adds the Equation 3.3 the change used at the previous iteration or time $(t-1)$ weighted by the momentum hyperparameter, as Equation 4.8 shows:

$$\Delta\theta(t) = \alpha \cdot \nabla f(\theta(t-1)) + momentum \cdot \Delta\theta(t-1) \tag{4.9}$$

In this project, a large momentum was used, which means the update of weights were strongly influenced by the previous update.

Another method used in the training configuration is gradient clipping (Pascanu et al., 2013). While calculating gradients of the parameters in order to converge the loss function, some of them may increase exponentially or "explode". This results in large updates to the network's learning parameters, making the network unstable. Gradient clipping method limits or clips gradients if their norm exceeds a certain threshold. This translates in the following equation, that states if a gradient $g$ is bigger than a threshold $u$ then $g$ value is set equal to the threshold value:

$$if\ \|g\| > u\ then\ g \leftarrow u \cdot \frac{g}{\|g\|} \tag{4.10}$$

Apart from gradient clipping, another widely-used method for ensuring a stable network was applied. This method is called L2 regularization that tunes the loss function by adding a penalty term to ensure that the trainable weights of the network do not excessively fluctuate. As Equation 4.11 shows, the penalty term is the norm of the weight vector multiplied by a lambda factor divided by 2, called weight decay, that determines the strength of the penalty.

$$L(w, b)_{new} = L(w, b) + \frac{\lambda}{2}\|w\|^2 \tag{4.11}$$

The general and training hyperparameter set for this project are displayed in Table 4.3.

| Hyperparameters | Values |
|---|---|
| Backbone network | resnet101 |
| Backbone strides | [4, 8, 16, 32] |
| Image shape | 1024×1024×3 |
| Mini mask shape | 56×56 |
| Learning rate | 0.0007 |
| Batch size | 1 |
| Learning momentum | 0.9 |
| Gradient clipping norm | 5.0 |
| Weight decay | 0.0001 |

Table 4.3: General configuration values of the Mask R-CNN algorithm.

## 4.5 Experimental phases and evaluation

The Mask R-CNN algorithm was used to create several models during the experimental process. In the first experimental phase, two models for litter detection are trained, one on the UAVVaste dataset and one on the CoastLitter dataset. In the second experimental phase, a Mask R-CNN model was trained on the CoastLitter dataset to detect litter and classifying it by material (plastic). In the third and last experimental phase, using the same dataset for training as with the second experimental phase, another Mask R-CNN model was created to categorize the detected litter by type (bottle, cup, can, other).

The models were evaluated in terms of average precision and recall metrics, famously used to evaluate object detectors in research and in competitions like the COCO and PASCAL VOC challenges. Mask R-CNN training and validation losses were monitored with TensorBoard (2019), which is TensorFlow's visualization toolkit. Predictions for each stage of the algorithm were evaluated with several visualizations like the precision-recall curve (section 3.6.4), predicted bounding boxes-ground-truth bounding boxes overlaps and more. The results are shown in the next chapter.

# Chapter 5

# Results

## 5.1 Internal model inspection

In this section, the prediction results at each stage of the Mask R-CNN for one of the trained models were visualized to get an idea of the model's internal workings. At the first stage of Mask R-CNN, the Region Proposal Network (RPN) generates anchors over the output images of the Feature Pyramid Network (FPN). Then, it returns a classification score for each of the anchors to categorize them into being an object or background. In other words, it classifies them as positive or negative. In order to make this classification, the RPN uses the ground-truth bounding boxes. The network assigns an anchor a positive label when the intersection over union (IoU) between the anchor and its ground-truth box exceeds 0.7. If the IoU is below 0.3, the network assigns the anchor a negative label (background). If the IoU is between 0.7 and 0.3, the anchor does not contribute to training. Figure 5.1a displays the top predicted positive anchors, sorted by their objectness score, for a sample image. The boundaries of the positive anchors may not cover entirely the objects. That is why RPN predicts 4 offset coordinates to be applied to the anchors to shift them to the correct boundaries of the objects. Figure 5.1b and 5.1c show the anchors refinement process.

RPN generates from all FPN levels 261888 anchors per image in total. After refinement the anchors are reduced to 6000. The non-maximum

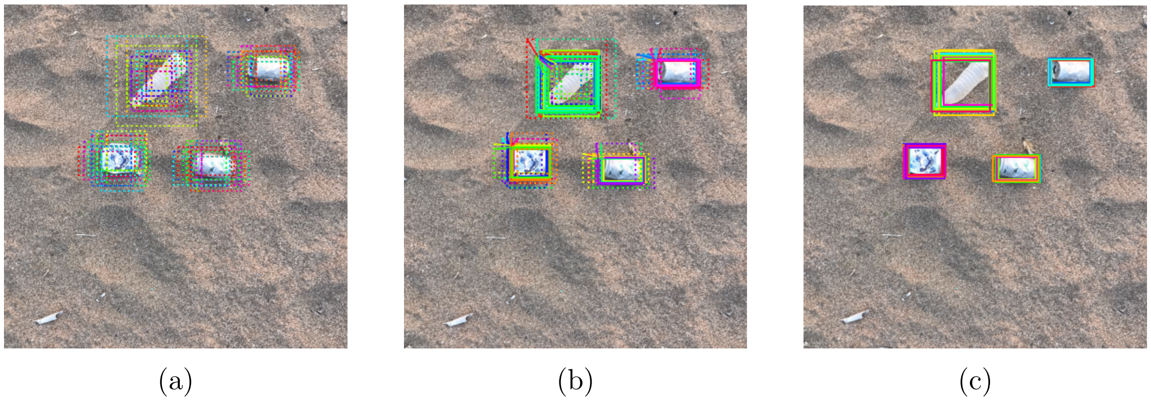


Figure 5.1: Anchors refinement.

suppression (NMS) algorithm filters the refined anchors searching for those that highly overlap and keeping the ones with the highest objectness score. In inference, the algorithm reduces effectively the number of anchors per image to 1000. These anchors are the final proposals and are passed to the stage two of the Mask R-CNN algorithm to be classified. A sample of these proposals after non-maximum suppression are displayed in Figure 5.2.

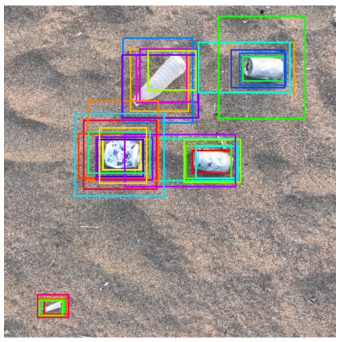


Figure 5.2: Anchors after NMS.

At the second stage of the Mask R-CNN, the model predicts the class probabilities and the bounding box offsets for each of the proposals. For the sample image, the model predicted 28 positive regions of interest out of 1000. It specifically classified 972 proposals as background, 9 as bottle, 18 as can and 1 as other. A random sample of the proposals is depicted in Figure 5.3a, where the dotted proposals are the ones classified as background and the rest have a label and a confidence score. Figure 5.3b shows the positive regions of interest after the refinement of the bounding boxes. Low-confidence region proposals were filtered out. The minimum detection confidence threshold for this model was set to 0.5. After this filtering process, 21 out of 28 proposals were kept. Specifically, they were kept 7 out of 9 instances classified as bottle and 14 out of 18 instances classified as can. In the sample image the proposal classified as other that has a confidence value of 0.482 was removed. Per-class non-maximum suppression was applied to the 21 regions of interest of the sample image. The NMS algorithm kept 1 out of the 7 instances labeled as bottle, and 3 out of the 14 instances labeled as can. Figure 5.3c displays the result of NMS. Figure 5.3d shows the refined final detections.

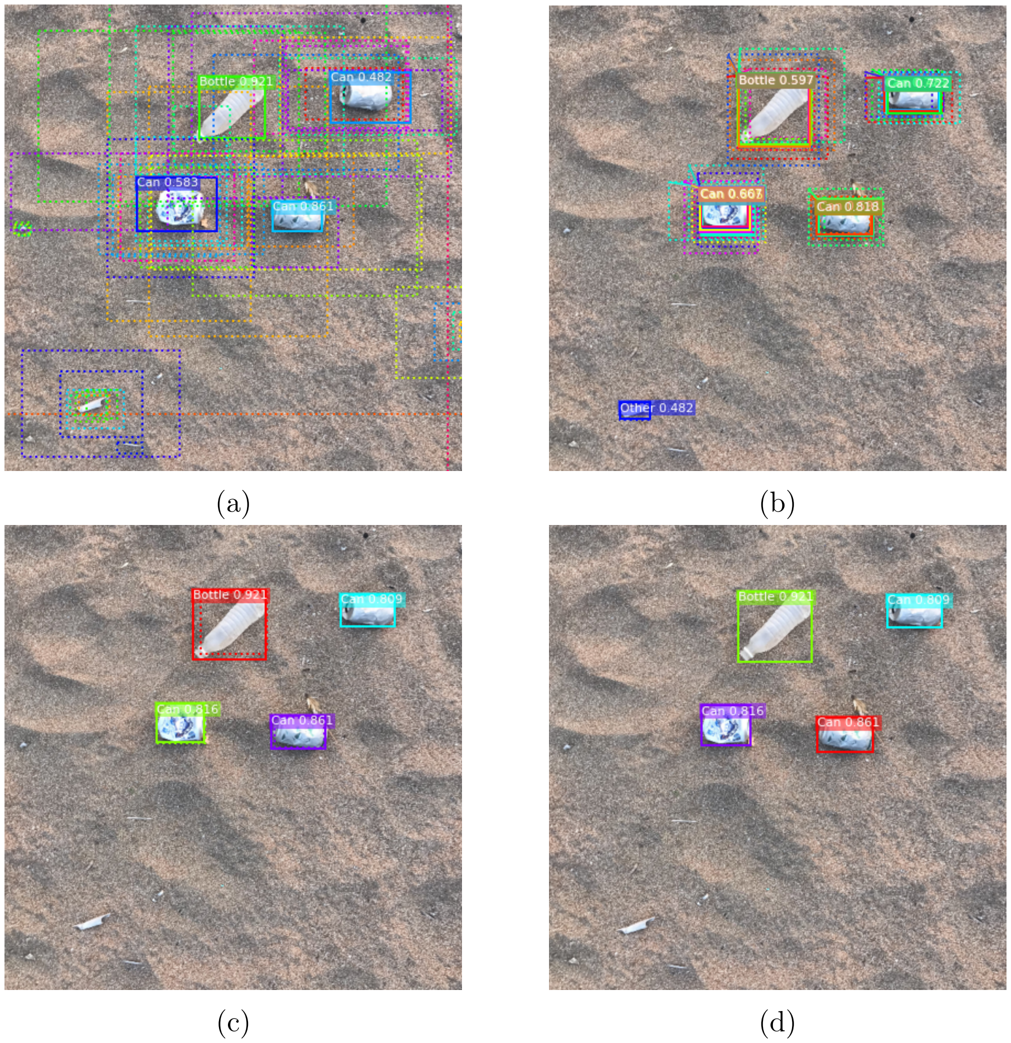
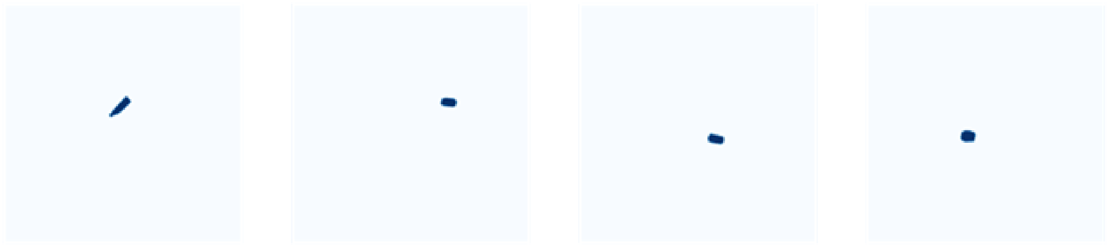


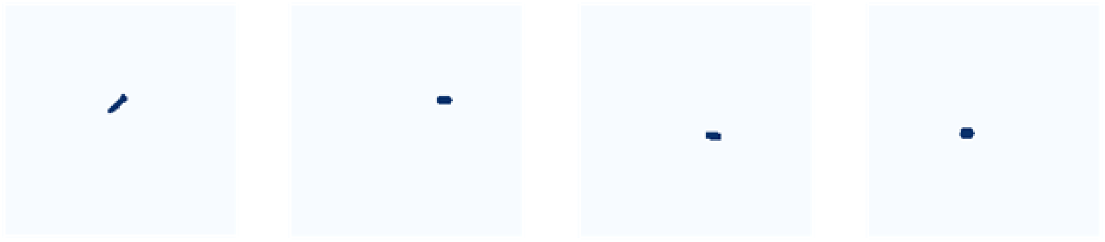
Figure 5.3: Region proposals refinement.

The refined detections are passed through the mask branch of the

Mask R-CNN, which generates segmentation masks for every detection. Figure 5.4a depicts the ground-truth masks and 5.4b shows the predicted masks of each detected instance. Finally, the predicted masks which have a size of 28 x 28, are scaled to the original image size of 1024 x 1024 and placed in the correct position, resulting in Figure 5.5.



(a)



(b)

Figure 5.4: Ground-truth masks vs predicted masks.

In the feature extraction stage of the Mask R-CNN algorithm, it is worth inspecting the output activation maps from different convolutional layers of the feature extractor network, here ResNet101, and see what features of the input are detected or preserved and look of odd signs and patterns. In Figure 5.6, each row of images is a sample of feature maps of a convolutional layer of the ResNet101 network. Feature maps are different versions of the original image with different features highlighted focusing either on the background or the foreground. Activations closer to the input are more fine-grained than the ones closer to the output of the extraction network. This is evident in Figure 5.6a, where it depicts a sample of activations of a convolutional layer near the beginning of the

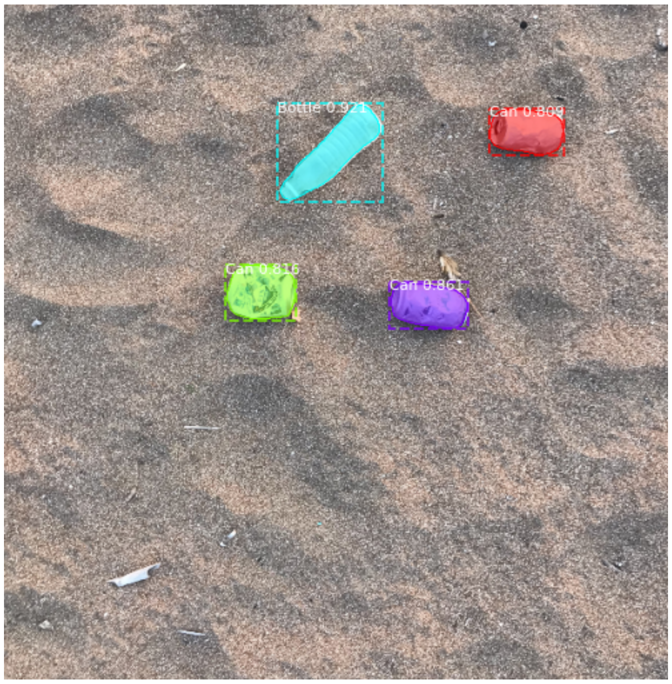
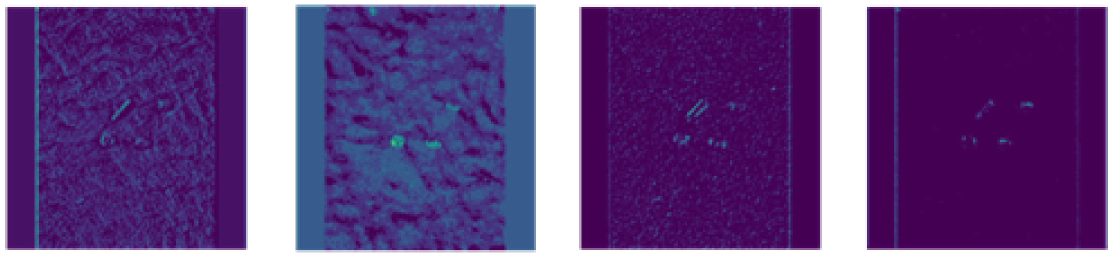
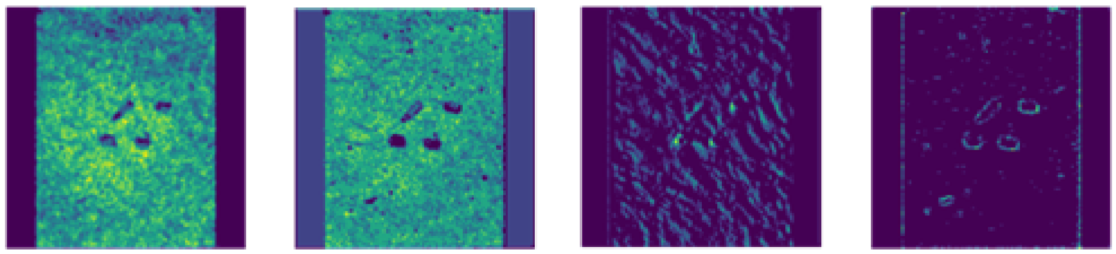


Figure 5.5: Final detection and segmentation result.

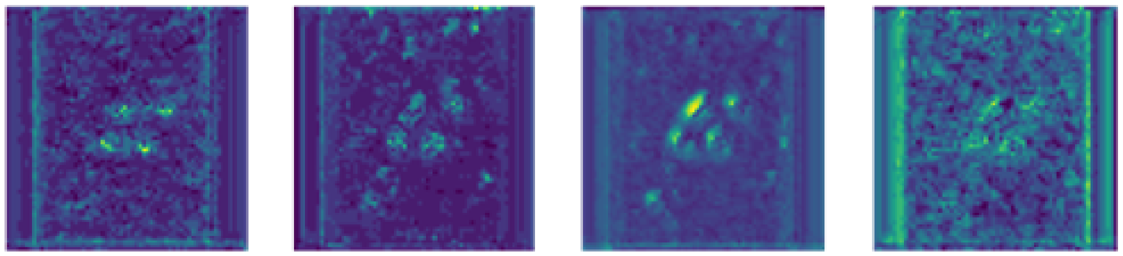
network, while Figure 5.6d depicts a sample of activations of one of the last layers of the network, hence the activations are more abstract.



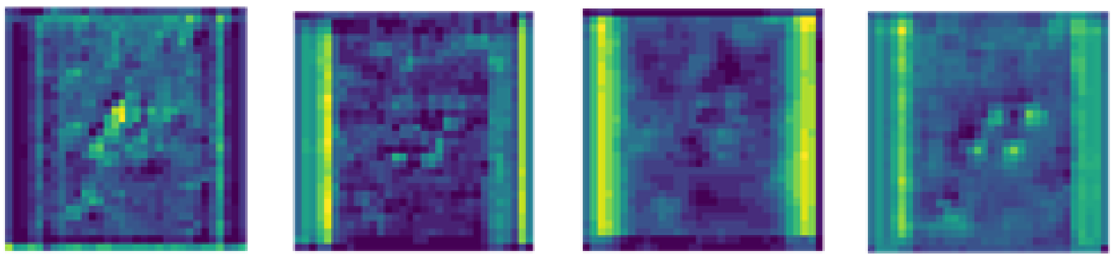
(a)



(b)



(c)



(d)

Figure 5.6: Activation maps of different layers of the feature extraction network.

## 5.2 Evaluation metrics

### 5.2.1 Model evaluation for UAVVaste and CoastLitter dataset

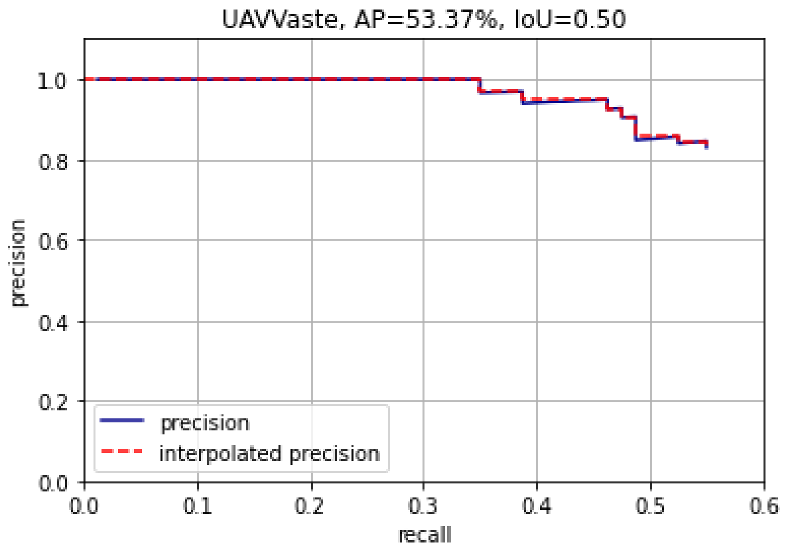
At the first experimental phase, two Mask R-CNN models were trained, one on the open-sourced UAVVaste dataset and one on the CoastLitter dataset. These models were trained to detect litter in the environment. This translates in having the models to localize and classify an object as litter or background, as well as generate a binary mask for each litter. The models were evaluated with the popular average precision (AP) metric that is the area under the precision-recall curve at a certain intersection over union (IoU) threshold. The model detections were ordered by their class confidence scores in a descending order. IoU values were calculated for each detection. The detections were classified as true positives if their IoU value was greater than the specified IoU threshold, otherwise they were classified as false positive. In case of multiple classes (not in this case), the detections were considered to be true positives whether their IoU value exceed the threshold and whether their class label matches the ground truth. The precision and recall values were estimated using Equation 3.9 and 3.10 for the accumulated true positive and false positive values of each detection. Then, the precision-recall curve was plotted to calculate AP as the area under the precision-recall curve. Here, AP was estimated over 10 IoU thresholds of .50:.05:.95 (COCO's primary challenge metric), at IoU=0.5 (PASCAL VOC's primary challenge metric), and at IoU=0.75 (strict metric). For each IoU threshold, AP was calculated using all-point interpolated precision values (Equation 3.22). Figures 5.7a and 5.7b show the precision and recall curves estimated using the actual precision values and the every-point interpolated precision values at IoU=0.5 and at IoU=0.75, respectively, for a batch of test images of the UAVVaste dataset.

The dependency of the AP metric on the IoU threshold is evident for the model trained on the UAVVaste dataset. The reduced performance of the model due to the use of a more restrictive IoU threshold value was expected because of the small size of the objects in aerial images. In contrast, the IoU threshold did not affect the model trained on the CoastLitter dataset because the detections had IoU values above the strict threshold of 0.75. Hence, the model had the same performance at IoU=0.5 and at IoU=0.75, as shown in Table 5.1. Figure 5.8 depicts the precision-recall curve estimated at IoU=0.5 for a batch of test images of the CoastLitter dataset.

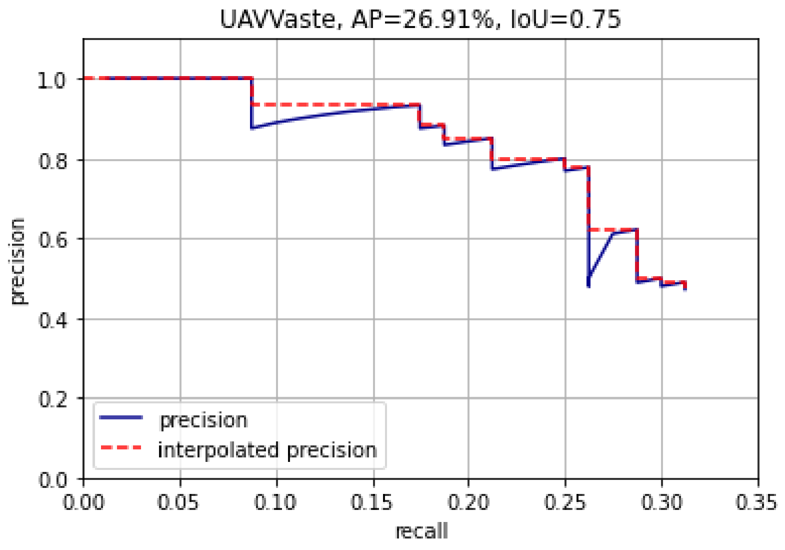
The model trained on the CoastLitter dataset exhibited very high performance with average precision, estimated at the IoU thresholds of 0.5 and 0.75, exceeding 90%. Looking at Table 5.1, for the model trained on the UAVVaste dataset, AP value at IoU=0.75 is approximately half the one calculated at IoU=0.5. The small scale of litter in aerial images reduces the accuracy of the model in locating litter, leading to IoU values less than the thresholds of 0.5 and 0.75, and thus resulting in the accumulation of false positive detections that reduce precision.

| | UAVVaste | CoastLitter |
|---|---|---|
| AP@.5 : .05 : .95 | 41.36 | 79.15 |
| AP@.50 | 53.37 | 92.43 |
| AP@.75 | 26.91 | 92.43 |

Table 5.1: Average precision of the models trained on UAVVaste and CoastLitter datasets.



(a)



(b)

Figure 5.7: Precision-recall curves at (a) IoU=0.5 and (b) IoU=0.75 of the model trained on UAVVaste dataset.

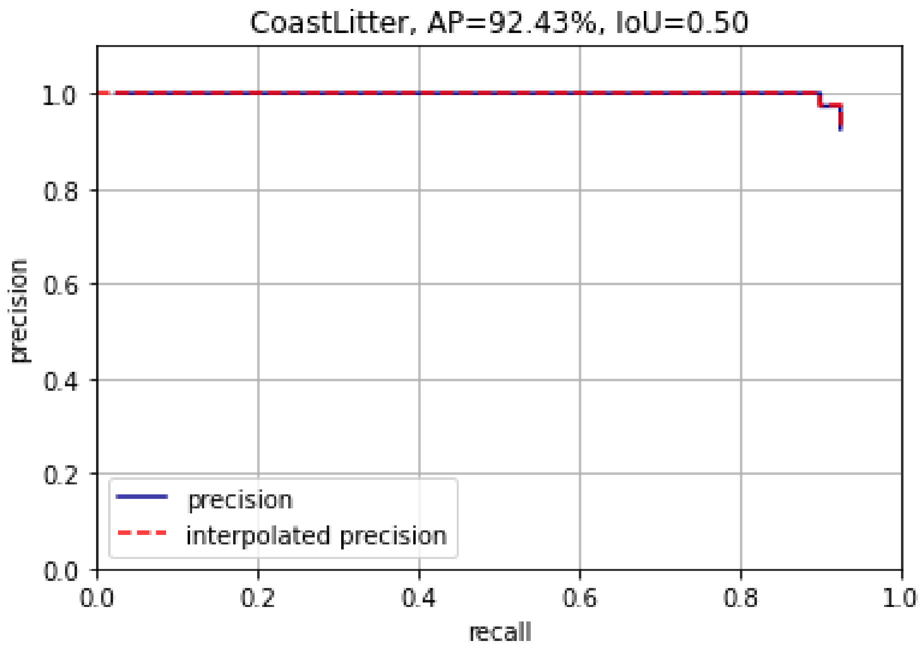


Figure 5.8: Precision-recall curve of the model trained on CoastLitter dataset.

### 5.2.2 Model evaluation for plastic and non-plastic litter categories

At this experimental phase, a Mask R-CNN model was trained to perform localization, classification by material, and instance segmentation on litter from the collected litter imagery. Since plastic litter is considered as a long-lasting threat to the environment, it was decided to train the model to detect and classify litter as either plastic or non-plastic. Contrary to the previous two models, this model had to classify a region proposal into one of 3 classes: plastic, non-plastic, or background. Since it was a little more complicated that the previous model, it was expected to have lower detection confidence in its predictions.

In the process of refining detections, after the proposals are classified, they are grouped by class and sorted according to their confidence scores before entering the NMS algorithm. The low confidence proposals are filtered out based on a threshold value. If the threshold value is set too high, more objects will be missed, resulting in more false negatives; on the contrary, if the setting is too low, more false positive detections will be made. The previous models exhibited very high confidence scores in their detections, thus the minimum confidence threshold was set to 0.9. Here, setting the threshold lower than 0.9 had a tremendous difference in average precision for both classes. Calculating average precision at IoU=0.5 at varying confidence thresholds for both classes results in Figure 5.9 that displays this difference.

AP was not estimated for a confidence threshold lower than 0.5 because of the accumulation of more false positive detections. Generally, a good value for the confidence threshold is one where both false positives and false negatives are low for all classes while maintaining a high AP. As shown in Figure 5.9, for both categories AP is the highest when the confidence threshold is set at 0.5. Furthermore, for this threshold value the number of false positive and false negative detections was low for both litter categories. Therefore, this value was retained for use in the estimation of the different AP metrics for each class. Again, AP was

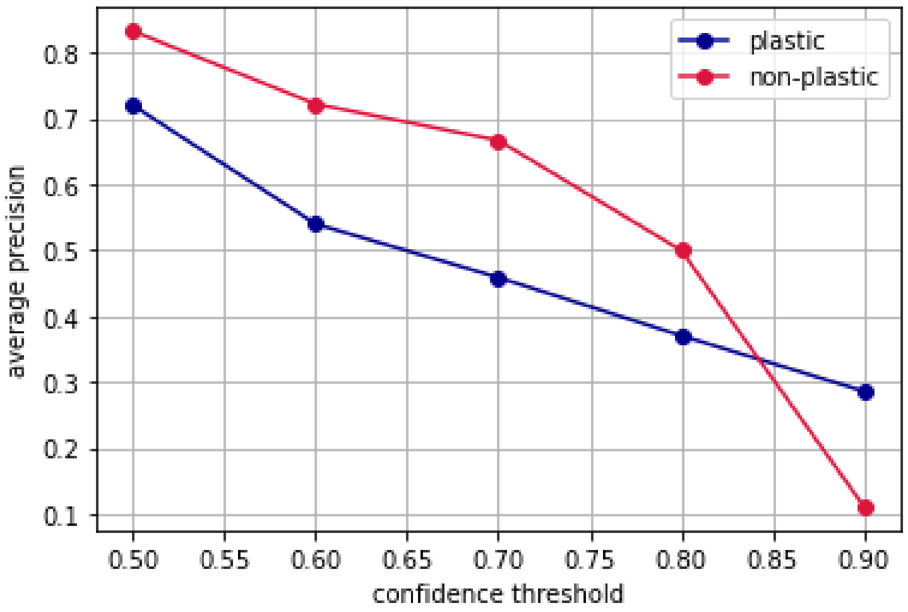


Figure 5.9: Average precision estimated for a range of confidence thresholds for plastic and non-plastic litter categories.

computed at 10 different IoU thresholds ranging from 50% to 95% at 5% step-size, at the commonly used IoU threshold of 0.5 and, at the restrictive IoU threshold of 0.75. Subsequently, the mean average precision (mAP), which is AP over all classes, was computed. Table 5.2 reports the results.

| | plastic | non-plastic | mAP |
|---|---|---|---|
| AP@.5 : .05 : .95 | 55.54 | 57.37 | 56.46 |
| AP@.50 | 72.03 | 83.30 | 77.67 |
| AP@.75 | 72.03 | 83.30 | 77.67 |

Table 5.2: Average precision for plastic and non-plastic litter categories.

As shown in Table 5.2, AP calculated at IoU=0.75 is the same with AP calculated at IoU=0.5, as the IoU values of the detections were greater than the thresholds. This is why Figures 5.10a and 5.10b show the precision-recall curves for each class estimated only at IoU=0.5. Again, AP was calculated as the area under the precision-recall curve using all-point interpolated precision values. The results of Table 5.2 show that in terms of average precision the model performed about 10% better at detecting non-plastic litter than detecting plastic litter. In the case of plastic class, there were more false positives because the model sometimes confused background materials with plastic objects. An example of this case is shown in Figure 5.11. AP calculated at IoU=[0.5:0.05:0.95] is quite similar for both classes and lower than the other AP metrics because higher IoU thresholds are taken into account, and therefore the precision decreases regardless of the litter category.

Inspecting the class balance shown in Figure 5.12, the plastic class is represented more than the non-plastic class. More instances of one class can affect the overall classification loss function in its favor. Depending on how big the difference is between the number of samples in each class, the one with the most samples will have a greater impact on the optimizer of the algorithm. In this case, the sample difference between the two classes does not seem to affect the accuracy in favor of the plastic class, as the AP is about 10% higher for the non-plastic class.

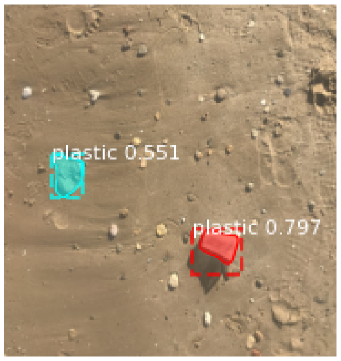
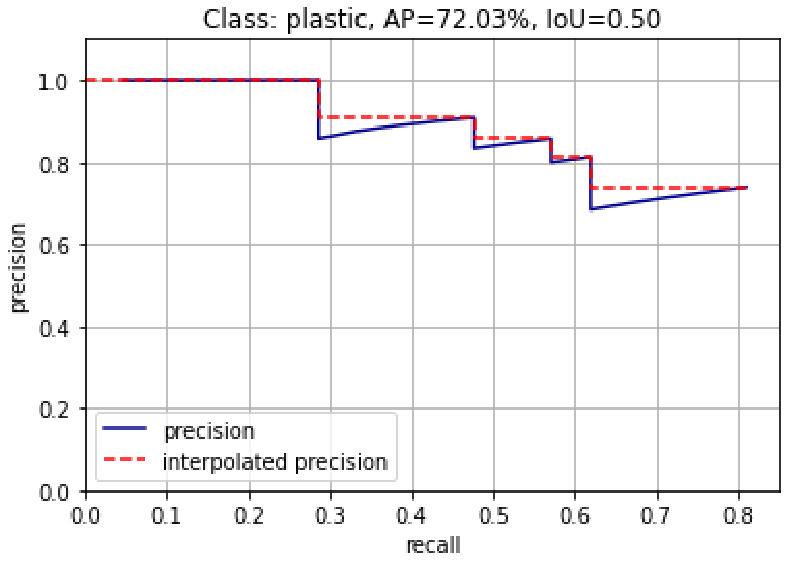
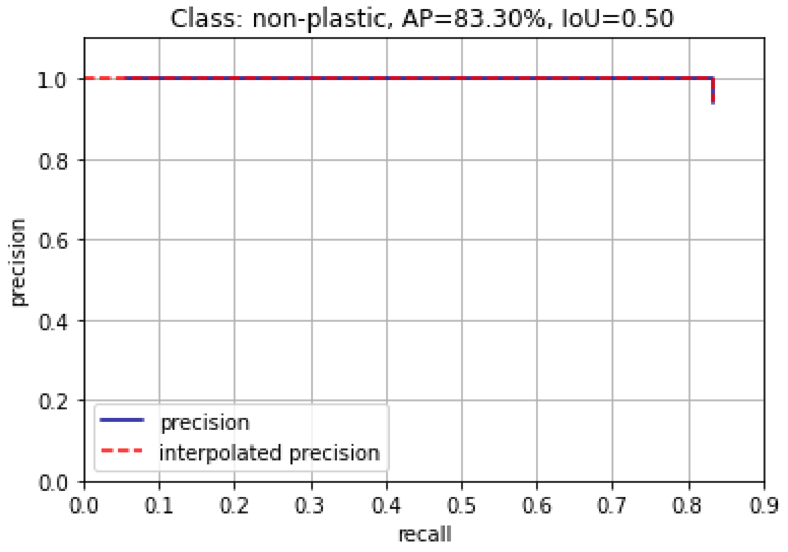


Figure 5.11: A sample image depicting a false (left item) and a correct detection (right item).



(a)



(b)

Figure 5.10: Precision-recall curves for (a) the plastic class and (b) the non-plastic class.

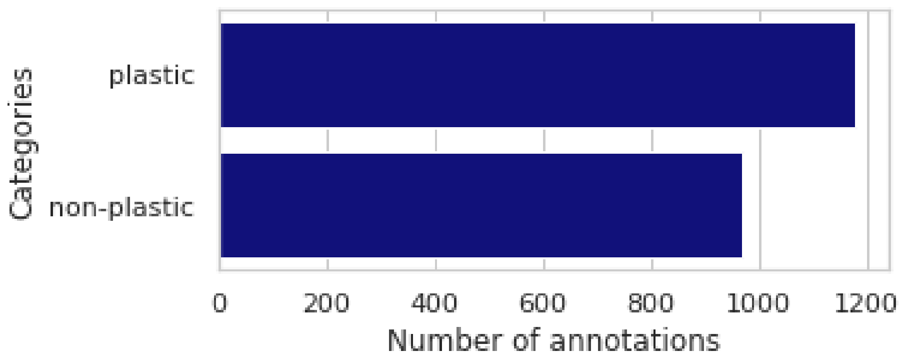


Figure 5.12: Number of litter annotations per class.

### 5.2.3 Model evaluation for litter categorized by type

At the last experimental phase, a Mask R-CNN model was trained to detect litter that belong to one of the 4 classes: bottle, cup, can, and other. When using the model for inference, several confidence thresholds were tested as with the previous model in order to determine the optimal value for fairly good detections. Figure 5.13 was created to exhibit the change in AP for each of the classes with respect to the different confidence threshold values.

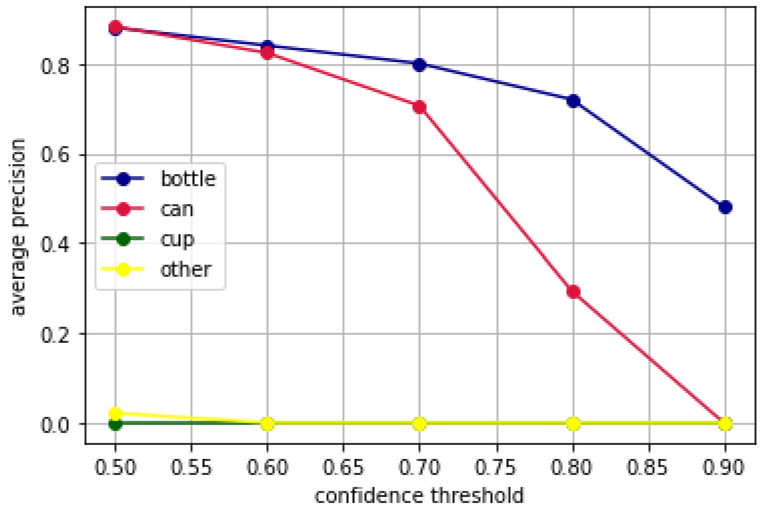


Figure 5.13: Average precision estimated for a range of confidence thresholds for four litter categories.

By lowering the threshold value from 0.9, AP increases for the bottle and can categories, but it stays flat for the other two. This means, that except for bottles and cans, the model failed to detect cups or other types of litter. In addition, for the cup category and the other category, it was observed that the model made several false predictions as the ones shown in Figure 5.14. Figure 5.14 shows a cup incorrectly labeled as a can and background material incorrectly labeled as other. Judging by the other two categories, the confidence threshold value chosen for the calculation of the different AP metrics was 0.5, due to the high value of AP, but also because of the low number of false positive and false

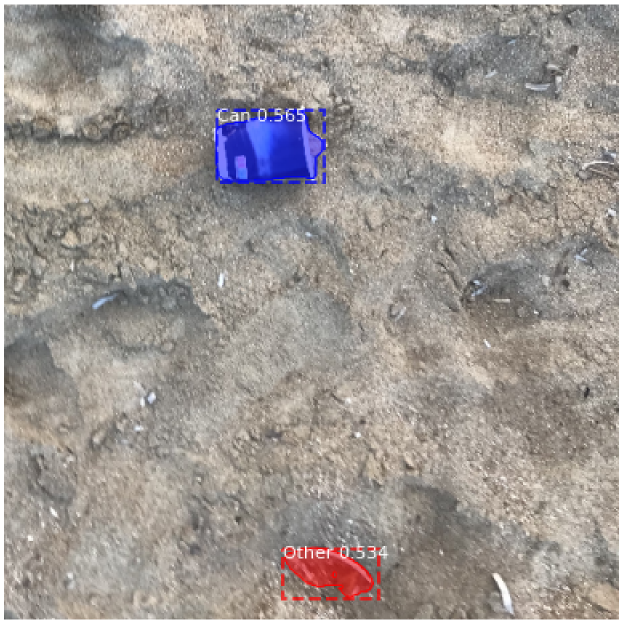


Figure 5.14: False predictions.

negative predictions. Referring to Figure 5.14, if the threshold value of 0.6 or higher was selected, the detections in Figure 5.14 would be false negatives (objects would not be detected) as the confidences are lower than 0.6.

In addition to the example in Figure 5.14, there were other cases where the model confused cups with cans, probably because they have similar shapes. The imbalance of annotated instances per class can be also a factor that can contribute to the model's false predictions especially if there are not many samples from each category. CoastLitter dataset is considered tiny for object detection and, as shown in Figure 5.15, there is an imbalance between the object classes. It can be seen that the cup category has the fewest litter samples, one reason that the model seemed more confident in labeling cups as cans. The number of samples in the category named as other is the highest but since it includes miscellaneous objects they were not enough for the model to learn to differentiate them from the other types of litter and the background. Also, the shape of litter in this category varies significantly, and it is considered to be the

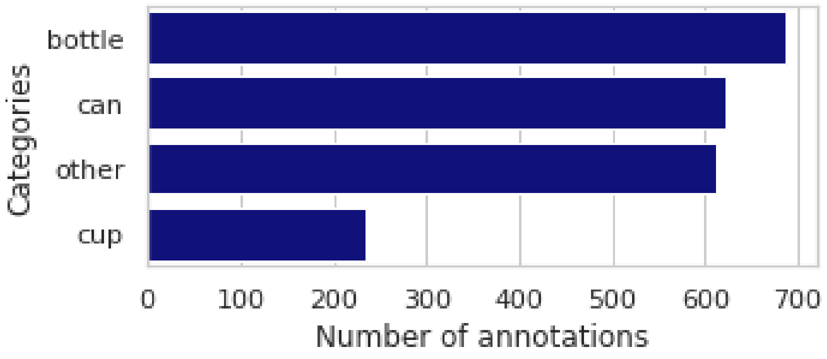


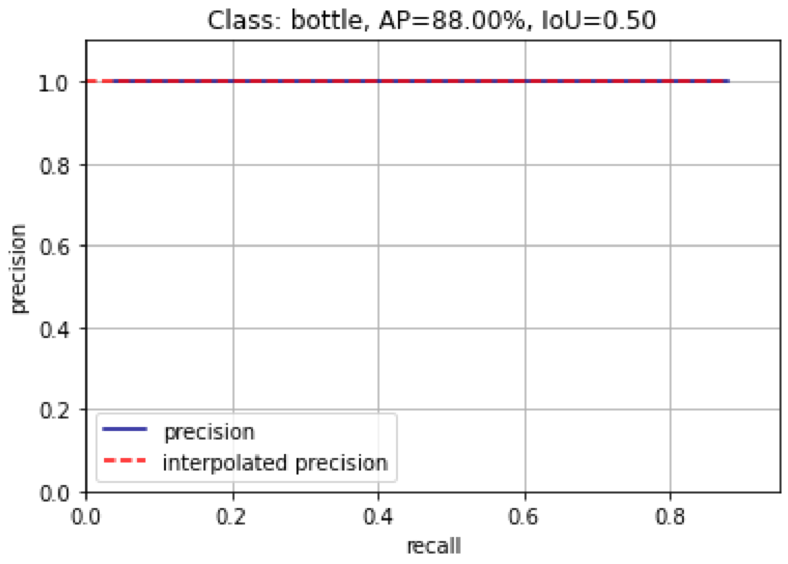
Figure 5.15: Number of litter annotations per class.

| | bottle | cup | can | other | mAP |
|---|---|---|---|---|---|
| AP@.5 : .05 : .95 | 77.99 | 0.00 | 66.41 | 1.25 | 36.41 |
| AP@.50 | 88.00 | 0.00 | 88.24 | 2.17 | 44.60 |
| AP@.75 | 88.00 | 0.00 | 88.24 | 0.53 | 44.19 |

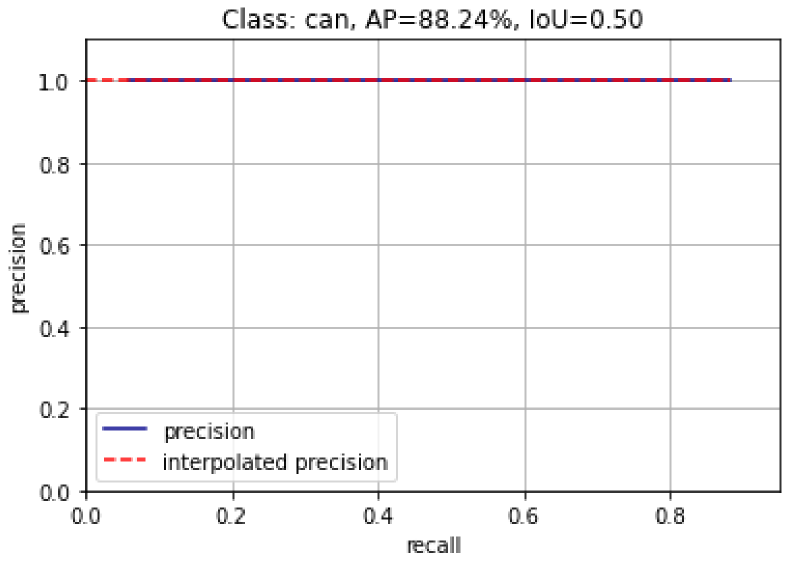
Table 5.3: Average precision for four litter categories.

reason why the model predicts incorrectly background objects (which also vary in shape) as litter. As for the bottle and can categories the model made good predictions. This is reflected in Table 5.3 where the calculations of the different AP metrics for each class are reported.

For the bottle and can categories, AP calculated at IoU=0.5 and at IoU = 0.75 is the same because the detections had IoU values higher than the IoU threshold of 0.75. AP averaged over the IoU threshold range of [0.5:0.05:0.95] is about 80% for the bottle class and 66% for the can class. Although, the other AP metrics showed that the model predicts cans slightly more accurately than bottles, it performs better on bottles when higher IoU thresholds are taken into account. This result may lie in the fact that the bottle class contains the highest number of samples as shown in Figure 5.15. AP values for the cup class and the other class show that the model fails to detect litter from these categories. Therefore, the precision-recall curves reported in this thesis are only for the bottle and can class shown in Figures 5.16a and 5.16b.



(a)



(b)

Figure 5.16: Precision-recall curves for (a) the bottle class and (b) the can class.

# Chapter 6

# Conclusions

In order to realize a deep learning based system of surveying litter from coastal areas, it is necessary to improve the accuracy of object detection on litter objects. This project explored the ability of Mask R-CNN algorithm to perform object detection and instance segmentation on coastline litter from images of a newly synthesized dataset, and also on litter from aerial images. Coastline litter was classified by material and by type to investigate the performance of the algorithm in differentiating in both cases.

Mask R-CNN demonstrated good accuracy and showed great potential in object detection and instance segmentation of litter in images from the CoastLitter dataset with average precision reaching a little over 90%. Comparing the accuracy of the algorithm on the images from the CoastLitter dataset to the one on aerial images from the UAVVaste dataset,the size factor was shown to affect the performance of the algorithm leading to lower accuracy in the case of aerial images. For the UAVVaste aerial images, AP calculated at the IoU threshold of 0.5 was about 53%, but when it was computed at the more restrictive threshold of 0.75, it dropped by half. This difference in values was also due to the small size of litter in UAVVaste images. This was not the case with the CoastLitter images where AP stayed the same for both IoU thresholds.

In the experimental case of predicting plastic over non-plastic litter, Mask R-CNN performed quite well having a mean average precision around 56-78% depending on which IoU threshold value was applied. The algorithm predicted non-plastic litter with slightly greater ease than plastic litter, even though the non-plastic class included less samples. Dependency on the minimum detection confidence threshold value was significant in this experimental phase. It is important to search for the appropriate threshold value which will lead to high detection accuracy, while retaining both false positive and false negative detections low.

Challenges were found when the algorithm had to classify litter into more categories. The difference in the performance of Mask R-CNN for each category was noticeable in the last experimental phase where it had to classify litter by type. The algorithm was effective in predicting bottles and cans with a maximum AP of around 88% for both litter categories. However, the algorithm failed to detect cups and other litter types. However, the algorithm failed to detect cups and other litter types. The difference in the number of litter samples for each of the four classes played a crucial role in the algorithm's predictions. The cup class was the least represented class among the others which led to the model's inability to predict cup samples. The model made several false positive detections on the category named as other. This category involves several types of litter that do not belong to the rest of the categories. To improve the performance of the model on under-represented classes or classes like the other category which involves miscellaneous types of objects, these classes have to be re-sampled. The cup class needs to be enriched with more instances to reinforce the model's performance on this class. Objects from the other category must be thoroughly examined, as they differ greatly in shape and size, to decide which types are under-represented and need adjustment in quantity. Another way to combat misclassification and low detection accuracy is to apply different weighted versions of cross-entropy classification loss to each object class (Phan and Yamamoto, 2020). Over-represented classes tend to be easy to predict because they have a greater impact on the overall classification loss than under-represented classes. Therefore, the gradient descent optimizes the detection for those majority classes. The performance of minority classes can be improved if a weight vector is applied to Equation 4.1 whose value is adjusted for each class. For a minority class, a larger value of w will increase the importance of the class during training. For an over-represented class that is relatively easily classified, a lower value of w will prevent its dominance in total loss.

Overall, the experiments demonstrated that the detection effectiveness relies heavily on the size of the objects, the selection of parameters such as the confidence threshold and the IoU threshold, as well as the number of samples for the different classes. Future work will focus on addressing the misclassifications occurred in the last experimental case by re-sampling the under-represented classes and adjusting the classification loss function to treat mistakes on these classes as more important than mistakes on well-represented classes. Other future work plans involve the extension of the coastline litter dataset and investigating the performance of other state-of-the-art object detection algorithms on it.

# Bibliography

Abdulla, W. (2017). Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. Github repository. https://github.com/matterport/Mask_RCNN.

Asensio-Montesinos, F., Anfuso, G., Randerson, P., and Williams, A. (2019). Seasonal comparison of beach litter on mediterranean coastal sites (Alicante, SE Spain). *Ocean and Coastal Management*, 181:104914. https://doi.org/10.1016/j.ocecoaman.2019.104914.

Barboza, L., Cózar, A., Gimenez, B., Barros, T., Kershaw, P., and Guilhermino, L. (2019). Macroplastics Pollution in the Marine Environment. *World Seas: an Environmental Evaluation*, pages 305–328. https://doi.org/10.1016/B978-0-12-805052-1.00019-X.

Bobulski, J. and Piątkowski, J. (2018). PET waste classification method and Plastic Waste DataBase WaDaBa. In *Conference Proceedings IP&C 2018, Advances in Intelligent Systems and Computing*, volume 681, pages 57–64. https://doi.org/10.1109/ICRA.2019.8793975.

Bochkovskiy, A., Wang, C., and Liao, H. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv*. https://arxiv.org/abs/2004.10934.

Bousquet, O., Boucheron, S., and Lugosi, G. (2004). Introduction to statistical learning theory. In Bousquet, O., von Luxburg, U., and Rätsch, G., editors, *Advanced Lectures on Machine Learning. ML 2003. Lecture Notes in Computer Science*, volume 3176, pages 169–207. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-28650-9_8.

Chen, J., Li, Q., Wang, H., and Deng, M. (2020). A Machine Learning Ensemble Approach Based on Random Forest and Radial Basis Function Neural Network for Risk Evaluation of Regional Flood Disaster: A Case Study of the Yangtze River Delta, China. *International Journal of Environmental Research and Public Health*, 17(1). 49. https://doi.org/10.3390/ijerph17010049.

Chen, L., Wu, C., Fan, W., Sun, J., and Naoi, S. (2014). Adaptive Local Receptive Field Convolutional Neural Networks for Handwritten Chinese Character Recognition. *Pattern Recognition*, pages 455–463. https://doi.org/10.1007/978-3-662-45643-9_48.

Chollet, F. (2017). *Deep Learning with Python*, chapter 1. Manning Publications.

Chuvieco, E. (2016). *Fundamentals of Satellite Remote Sensing: An Environmental Approach, Second Edition.* CRC Press.

Cooper, M. J. (2018). *A Deep Learning Prediction Model for Mortgage Default.* Master's thesis, University of Bristol, England. http://dx.doi.org/10.13140/RG.2.2.21506.12487.

Dhingra, G., Kumar, V., and Joshi, H. D. (2019). A novel computer vision based neutrosophic approach for leaf disease identification and classification. *Measurement*, 135:782–794. https://doi.org/10.1016/j.measurement.2018.12.027.

Dutta, A. and Zisserman, A. (2019). The VIA Annotation Software for Images, Audio and Video. *arXiv*. https://arxiv.org/abs/1904.10699.

Erni-Cassola, G., Zadjelovic, V., Gibson, M. I., and Christie-Oleza, J. A. (2019). Distribution of plastic polymer types in the marine environment; A meta-analysis. *Journal of Hazardous Materials*, 369:691–698. https://doi.org/10.1016/j.jhazmat.2019.02.067.

Everingham, M., Eslami, S., Gool, L. V., Williams, C., Winn, J., and Zisserman, A. (2015). The PASCAL Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 111:98–136. https://doi.org/10.1007/s11263-014-0733-5.

Fallati, L., Polidori, A., Salvatore, C., Saponari, L., Savini, A., and Galli, P. (2019). Anthropogenic Marine Debris assessment with Unmanned Aerial Vehicle imagery and deep learning: A case study along the beaches of the Republic of Maldives. *Science of The Total Environment*, 693:133581. https://doi.org/10.1016/j.scitotenv.2019.133581.

Fulton, M., Hong, J., Islam, M. J., and Sattar, J. (2019). Robotic detection of marine litter using deep visual detection models. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5752–5758. https://doi.org/10.1109/ICRA.2019.8793975.

Galgani, L., Beiras, R., Galgani, F., Panti, C., and Borja, A. (2019). Editorial: Impacts of Marine Litter. *Frontiers in Marine Science*, 6. 10.3389/fmars.2019.00208.

Geraeds, M., van Emmerik, T., de Vries, R., and bin Ab Razak, M. S. (2019). Riverine Plastic Litter Monitoring Using Unmanned Aerial Vehicles (UAVs). *Remote Sensing*, 11(17):2045. https://doi.org/10.3390/rs11172045.

Girshick, R. (2015). Fast R-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448. https://doi.org/10.1109/ICCV.2015.169.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587. https://doi.org/10.1109/CVPR.2014.81.

Gonçalves, G., Andriolo, U., Gonçalves, L., Sobral, P., and Bessa, F. (2020). Quantifying Marine Macro Litter Abundance on a Sandy Beach using Unmanned Aerial Systems and Object-Oriented Machine Learning Methods. *Remote Sensing*, 12(16):2599. https://doi.org/10.3390/rs12162599.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Google Research (2019). Open images 2019 - object detection: Detect objects in varied and complex images. https://www.kaggle.com/c/open-images-2019-object-detection.

Goëau, H., Mora-Fallas, A., Champ, J., Rossington-Love, N. L., Mazer, S. J., Mata-Montero, E., Joly, A., and Bonnet, P. (2020). A new fine-grained method for automated visual analysis of herbarium specimens: A case study for phenological data extraction. *Applications in Plant Sciences*, 8(6). https://doi.org/10.1002/aps3.11368.

Hartmann, N. B., Hüffer, T., Thompson, R. C., Hassellöv, M., Verschoor, A., Daugaard, A. E., Rist, S., Karlsson, T., Brennholt, N., Cole, M., Herrling, M. P., Hess, M. C., Ivleva, N. P., Lusher, A. L., and Wagner, M. (2019). Are We Speaking the Same Language? Recommendations for a Definition and Categorization Framework for Plastic Debris. *Environmental Science & Technology*, 53(3):1039–1047. https://doi.org/10.1021/acs.est.8b05297.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988. https://doi.org/10.1109/ICCV.2017.322.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. https://doi.org/10.1109/CVPR.2016.90.

Hong, J., Fulton, M., and Sattar, J. (2020a). A Generative Approach Towards Improved Robotic Detection of Marine Litter. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10525–10531. https://doi.org/10.1109/ICRA40945.2020.9197575.

Hong, J., Fulton, M., and Sattar, J. (2020b). TrashCan: A Semantically-Segmented Dataset towards Visual Detection of Marine Debris. *arXiv*. https://arxiv.org/abs/2007.08097.

Hosang, J., Benenson, R., Dollár, P., and Schiele, B. (2016). What makes for effective detection proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):814–830. https://doi.org/10.1109/TPAMI.2015.2465908.

Kraft, M., Piechocki, M., Ptak, B., and Walas, K. (2021). Autonomous, Onboard Vision-Based Trash and Litter Detection in Low Altitude Aerial Images Collected by an Unmanned Aerial Vehicle. *Remote Sensing*, 13(5):965. https://doi.org/10.3390/rs13050965.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc. https://doi.org/10.1145/3065386.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature Pyramid Networks for Object Detection. *arXiv*. https://arxiv.org/abs/1612.03144v2.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. (2015). Microsoft COCO: Common Objects in Context. *arXiv*. https://arxiv.org/abs/1405.0312.

Lo, H.-S., Wong, L.-C., Kwok, S.-H., Lee, Y.-K., Po, B. H.-K., Wong, C.-Y., Tam, N. F.-Y., and Cheung, S.-G. (2020). Field test of beach litter assessment by commercial aerial drone. *Marine Pollution Bulletin*, 151:110823. https://doi.org/10.1016/j.marpolbul.2019.110823.

Long, J., Shelhamer, E., and Darrell, T. (2014). Fully Convolutional Networks for Semantic Segmentation. *arXiv*. http://arxiv.org/abs/1411.4038.

Maitre, J., Bouchard, K., and Bédard, L. P. (2019). Mineral grains recognition using computer vision and machine learning. *Computers and Geosciences*, 130:84–93. https://doi.org/10.1016/j.cageo.2019.05.009.

MARLIN (2013). Final report of Baltic marine litter project MARLIN - Litter Monitoring and Raising Awareness 2011–2013. http://pidasaaristosiistina.fi/files/1994/Marlin_Final_Report_2014.pdf.

Martin, C., Parkes, S., Zhang, Q., Zhang, X., McCabe, M. F., and Duarte, C. M. (2018). Use of unmanned aerial vehicles for efficient beach litter monitoring. *Marine Pollution Bulletin*, 131:662–673. https://doi.org/10.1016/j.marpolbul.2018.04.045.

Martínez-Vicente, V., Clark, J. R., Corradi, P., Aliani, S., Arias, M., Bochow, M., Bonnery, G., Cole, M., Cózar, A., Donnelly, R., Echevarría, F., Galgani, F., Garaba, S. P., Goddijn-Murphy, L., Lebreton, L., Leslie, H. A., Lindeque, P. K., Maximenko, N., Martin-Lauzer, F.-R., Moller, D., Murphy, P., Palombi, L., Raimondi, V., Reisser, J., Romero, L., Simis, S. G., Sterckx, S., Thompson, R. C., Topouzelis, K. N., van Sebille, E., Veiga, J. M., and Vethaak, A. D. (2019). Measuring Marine Plastic Debris from Space: Initial Assessment of Observation Requirements. *Remote Sensing*, 11(20):2443. https://doi.org/10.3390/rs11202443.

Merlino, S., Paterni, M., Berton, A., and Massetti, L. (2020). Unmanned Aerial Vehicles for Debris Survey in Coastal Areas: Long-Term Monitoring Programme to Study Spatial and Temporal Accumulation of the Dynamics of Beached Marine Litter. *Remote Sensing*, 12(8):1260. https://doi.org/10.3390/rs12081260.

NVIDIA Developer (2020). Jetson Xavier NX Developer Kit. https://developer.nvidia.com/embedded/jetson-xavier-nx-devkit.

Nwankpa, C. E., Ijomah, W., Gachagan, A., and Marshall, S. (2018). Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. *arXiv*. https://arxiv.org/pdf/1811.03378.pdf.

Olivelli, A. and Rosebrock, U. (2020). From Hobart, to London, to Dhaka: using cameras and AI to build an automatic litter detection system. https://theconversation.com/from-hobart-to-london-to-dhaka-using-cameras-and-ai-to-build-an-automatic-litter-detection-system-150950.

OSPAR (2010). Guideline for Monitoring Marine Litter on the Beaches in the Ospar Maritime Area. https://www.ospar.org/ospar-data/10-02e_beachlitter%20guideline_english%20only.pdf.

Padilla, R., Netto, S. L., and da Silva, E. A. B. (2020). A survey on performance metrics for object-detection algorithms. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 237–242. http://doi.org/10.1109/IWSSIP48289.2020.9145130.

Pascanu, S., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28(3), pages 1310–1318. http://proceedings.mlr.press/v28/pascanu13.pdf.

Pervez, R., Wang, Y., Mahmood, Q., Zahir, M., and Jattak, Z. (2020). Abundance, type, and origin of litter on No. 1 Bathing Beach of Qingdao, China. *Journal of Coastal Conservation*, 24:34. https://doi.org/10.1007/s11852-020-00751-x.

Phan, T. H. and Yamamoto, K. (2020). Resolving Class Imbalance in Object Detection with Weighted Cross Entropy Losses. *arXiv*. https://arxiv.org/abs/2006.01413.

Ping, P., Kumala, E., Gao, J., and Xu, G. (2020). Smart street litter detection and classification based on Faster R-CNN and edge computing. *Journal of Software Engineering and Knowledge Engineering*, 30(4):537–553. https://doi.org/10.1142/S0218194020400045.

Proença, P. F. and Simões, P. (2020). TACO: Trash Annotations in Context for Litter Detection. *arXiv*, abs/2003.06975. https://arxiv.org/abs/2003.06975.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. *arXiv*. https://arxiv.org/abs/1506.02640.

Redmon, J. and Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *arXiv*. http://arxiv.org/abs/1804.02767.

Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031.

Rezatofighi, S. H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I. D., and Savarese, S. (2019). Generalized intersection over union: A metric and A loss for bounding box regression. *arXiv*. http://arxiv.org/abs/1902.09630.

Robertson, S., Azizpour, H., and Hartman, K. S. J. (2018). Digital image analysis in breast pathology-from image processing techniques to artificial intelligence. *Translational Research: The Journal of Laboratory and Clinical Medicine*, 194:19–35. https://doi.org/10.1016/j.trsl.2017.10.010.

Ruder, S. (2017). An overview of gradient descent optimization algorithms. *arXiv*. https://arxiv.org/abs/1609.04747.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115:211–252. https://doi.org/10.1007/s11263-015-0816-y.

Ruthotto, L. and Haber, E. (2021). An Introduction to Deep Generative Modeling. *arXiv*. https://arxiv.org/abs/2103.05180.

Sahni, S., Mittal, A., Kidwai, F., Tiwari, A., and Khandelwal, K. (2020). Insurance Fraud Identification using Computer Vision and IoT: A Study of Field Fires. *Procedia Computer Science*, 173:56–63. https://doi.org/10.1016/j.procs.2020.06.008.

Schwartzman, A., Kagan, M., Mackey, L., Nachman, B., and Oliveira, L. D. (2016). Image Processing, Computer Vision, and Deep Learning: new approaches to the analysis and physics interpretation of LHC events. *Journal of Physics: Conference Series*, 762:012035. https://doi.org/10.1088/1742-6596/762/1/012035.

Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv*. http://arxiv.org/abs/1409.1556.

Stanford University (n.d.). Neural networks part 1, setting up the architecture. *CS231n Convolutional Neural Networks for Visual Recognition*. https://cs231n.github.io/neural-networks-1/.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9. https://doi.org/10.1109/CVPR.2015.7298594.

Tan, M., Pang, R., and Le, Q. V. (2019). EfficientDet: Scalable and Efficient Object Detection. *arXiv*, abs/1911.09070. http://arxiv.org/abs/1911.09070.

TensorBoard (2019). TensorBoard: TensorFlow's visualization toolkit. Google. https://www.tensorflow.org/tensorboard.

Thung, G. (2016). GitHub - garythung/trashnet: Dataset of images of trash; Torch-based CNN for garbage image classification. https://developer.nvidia.com/embedded/jetson-xavier-nx-devkit.

Ting, K. M. (2011). Precision and Recall. In Sammut, C. and Webb, G., editors, *Encyclopedia of Machine Learning.*, page 781. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30164-8_652.

Traore, B. B., Kamsu-Foguem, B., and Tangara, F. (2018). Deep convolution neural network for image recognition. *Ecological Informatics*, 48:257–268. https://doi.org/10.1016/j.ecoinf.2018.10.002.

Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171. https://doi.org/10.1007/s11263-013-0620-5.

Ulhaq, A., Khan, A., Gomes, D., and Paul, M. (2020). Computer vision for COVID-19 control: A survey. *arXiv*, pages 1–24. https://doi.org/10.31224/osf.io/yt9sx.

Ventura, D., Bonifazi, A., Gravina, M. F., Belluscio, A., and Ardizzone, G. (2018). Mapping and Classification of Ecologically Sensitive Marine Habitats Using Unmanned Aerial Vehicle (UAV) Imagery and Object-Based Image Analysis (OBIA). *Remote Sensing*, 10(9):1331. https://doi.org/10.3390/rs10091331.

Zhang, J., Song, X., Feng, J., and Fei, J. (2021). X-Ray Image Recognition Based on Improved Mask R-CNN Algorithm. *Mathematical Problems in Engineering*, 2021. https://doi.org/10.1155/2021/6544325.