LUISS

Data Science and Management

Course of Machine Learning

Enhancing maritime Search and Rescue with Small Object Detection techniques

Supervisor:

Giuseppe F. Italiano

Co-supervisor:

Alessio Brancati, Emilio Coppa

Candidate:

Francesco Del Treste

783351

Academic Year 2024/2025

Dedica

Abstract

The growing complexity of maritime Search and Rescue (SAR) operations highlights the need for advanced technological tools capable of supporting the detection of people at sea. In this context, this thesis investigates the use of deep learning techniques for real-time object detection, with the dual objective of meeting immediate industrial needs while fostering exploratory research directions.

The experimental work has confirmed that YOLO-based architectures remain the most effective solutions when real-time performance is required. Beyond these established models, the thesis introduces MaYOLOc, a novel YOLO-based variant that integrates contextual information through an attention mechanism. Unlike state-of-the-art approaches, which typically treat detections as isolated instances, MaYOLOc explicitly leverages scene context to guide learning dynamics and improve robustness in challenging maritime conditions. The goal of MaYOLOc is not to establish a definitive benchmark but to be a proof-of-concept to explore the feasibility of integrating contextual information into the detection pipeline. Although preliminary results reveal clear limitations due to data and training constraints, they also highlight the potential benefits of this approach, laying the basis for deeper research.

The results indicate that the most effective strategy lies in a two-phase approach. In the short term, a fine-tuned YOLOv11 model provides an operationally reliable solution, enabling Leonardo to deploy a robust system in real-world SAR contexts. In parallel, the progressive refinement of the MaYOLOc model, supported by the continuous collection of additional data, offers a pathway toward exploiting contextual information to further improve detection robustness. The encouraging signals observed during the explainability phase strengthen the rationale for pursuing this line of investigation, despite the current limitations of the prototype. Ultimately, the contributions presented in this thesis should be regarded not as a final outcome but as a foundation for sustained innovation, combining immediate industrial applicability with long-term research perspectives.

Acknowledgments

This thesis was conducted at Leonardo Electronics, within the Innovation and Digital Transformation (I&DT) laboratory of the AI and Big Data team. I would like to express my deepest gratitude to Ph.D. Christian Giusti, team manager, and Dr. Alessio Brancati, my supervisor, for their invaluable mentorship and support throughout this work. I am also sincerely grateful to all the team members and thesis colleagues for their encouragement, collaboration, and for creating such a positive and stimulating work environment. Their presence made this journey truly inspiring and enjoyable.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

While it is true that data have become one of the most valuable assets in recent years, its importance is also and especially due to its role as a fuel for advanced computational models designed to tackle real-world challenges. In high-tech environments, the integration of data-driven approaches with artificial intelligence has proven to be a transformative force, particularly in scenarios where human lives are at stake.

As this thesis will demonstrate, the application of deep learning to challenges where human lives are at risk is not only essential for improving the accuracy of mission-critical objectives, but also plays a crucial role in supporting operators by reducing cognitive load and mitigating the risk of human fatigue in repetitive tasks. Furthermore, implementing these techniques in real-world scenarios significantly enhances the promptness of intervention, thereby increasing the likelihood of saving lives.

In the chapters that follow, this thesis will specifically analyze the critical challenge of locating individuals in maritime emergency situations, where unpredictable environmental conditions, vast search areas, and time-sensitive operations make rescue efforts particularly complex. Factors such as extreme weather, limited visibility, and the rapid onset of hypothermia in open water significantly reduce the chances of survival, emphasizing the need for swift and precise intervention. Addressing these challenges requires innovative technological solutions capable of enhancing situational awareness and optimizing search and rescue missions.

To this end, from a technical perspective, this will be achieved through the development of an advanced and highly sensitive visual model capable of detecting even the smallest details essential for identifying human presence during shipwrecks, storms, and other extreme conditions. The following sections will provide an exploration of the project and its stakeholders, the current landscape of the maritime search and rescue sector and the objectives set for this thesis.

1.1 Motivation for the thesis

The findings of this research stem from my work at Leonardo S.p.A[1], where I was actively involved in a project developed for the Italian Coast Guard. As part of this initiative, I contributed to the design and implementation of advanced technological solutions aimed at enhancing maritime search and rescue operations. This experience provided valuable insights into real-world operational challenges and significantly influenced the methodological approach adopted in this thesis.

My contributions to this work encompass a broad range of tasks, including the generation and the validation of a proper dataset from raw video data, the labeling of those data to feed the model, as well as its actual development. The opportunity to leverage Leonardo's extensive resources, including its advanced laboratories and expertise in this field, has been invaluable to this research. Additionally, I had the privilege of utilizing Leonardo's state-of-the-art HPC[2], Davinci, to perform heavily computational training tasks. Access to these resources has significantly enhanced the efficiency and depth of the computational processes, contributing to the robustness of the developed model.

[1]Henceforth referred to as "Leonardo"
[2]High Performance Computing

As stated by the company itself, ”Leonardo is a global industrial group that builds technological capabilities in Aerospace, Defence & Security. The company plays a prominent role in major international strategic programmes and is a trusted technological partner of governments, defence agencies, institutions, and enterprises.” This definition underscores the company’s pivotal role within the security landscape and, consequently, the relevance of this thesis. The significance of Leonardo’s contributions to technological advancements in defence and security further justifies the focus of this research, highlighting the broader implications of its innovations in these critical domains. The Italian Guard Coast is a specialized branch of the Italian Navy responsible for maritime safety, search and rescue, environmental protection, and fisheries enforcement. Established in 1865, it operates under the Ministry of Infrastructure and Transport, with additional duties assigned by other governmental bodies. With 11,000 personnel and a widespread network of offices, it ensures security and regulatory oversight at sea and along the Italian coastline. The high-level collaboration between these two stakeholders has led to the development of a cutting-edge, safety-centric technology.

1.2 Context of the maritime SAR sector

In Italy, maritime search and rescue (SAR) operations fall within a well-defined legal and organisational framework, centrally coordinated by the Italian Coast Guard (*Corpo delle Capitanerie di porto – Guardia Costiera*). Under the terms of the National SAR Plan, the Coast Guard is vested with the primary authority and mandate to conduct and supervise all SAR-related activities in Italian territorial waters.

At the operational level, the Maritime Rescue Coordination Centre (MRCC) in Rome, managed by the Coast Guard, is charged with receiving and evaluating distress signals, deploying appropriate resources, and orchestrating rescue missions. While the Coast Guard retains the legal mandate for overall coordination and control, it frequently cooperates with other governmental entities, such as the Italian Navy, local police forces, and the Fire Corps, as well as with non-governmental organizations, especially in large-scale or complex rescue scenarios. Nevertheless, the centralized responsibility and oversight for maritime SAR within Italy remain unequivocally under the jurisdiction of the Coast Guard, in line with domestic legislation and international obligations.

The Italian MRCC is in charge of the SAR duty within a specific SAR zone (Fig. 1.1), a maritime geographical area for which a particular state is responsible for coordinating search and rescue operations. These zones are defined by the International Convention on Maritime Search and Rescue (SAR Convention), adopted by the International Maritime Organisation (IMO) in 1979. The main objective of SAR zones is to ensure a global system of coordination to assist people in distress at sea as efficiently and promptly as possible. Each SAR zone is defined by specific boundaries and is assigned to a state that has the obligation to intervene and coordinate assistance in the event of maritime emergencies.

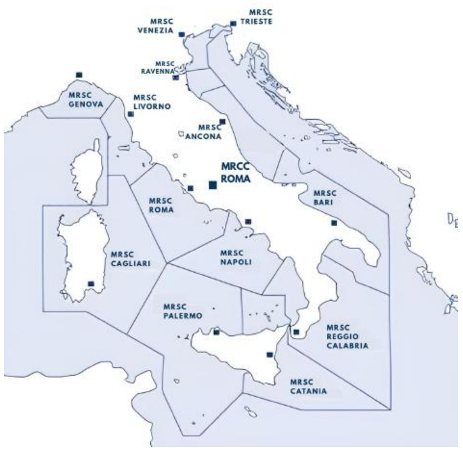


Fig. 1.1: Italian SAR zones

As illustrated in Figure 1.1, in Italy each SAR zone is further subdivided into smaller areas managed by Maritime Rescue Secondary Centres (M.R.S.C.) situated in major port cities throughout the peninsula. This organizational structure enhances both the efficiency and the responsiveness of emergency interventions.

Shifting from organizational and bureaucratic considerations to focus on technological and operational aspects, the Coast Guard has a fleet of vessels supported by various technologies to ensure maritime safety. In particular, it has a specific component for the maritime SAR, known in Italian as the *Componente ricerca e soccorso in mare*, (Search and Rescue Component). Other components of the fleet include the Offshore Component, Fisheries Surveillance Component, Coastal Component, Littoral (Nearshore) Component, and Logistics/Auxiliary Component, each fulfilling distinct operational roles within the broader mission of maritime protection and regulation. Their combined efforts uphold national and international regulations, protect marine ecosystems, and guarantee swift responses to emerging threats or urgent needs. These operational units utilize a broad range of technologies, which are briefly summarized in Table 1.1.

Table 1.1: Overview of maritime monitoring technologies of the Italian Coast Guard

Name of Technology	**Main Application**	**Description**
VTS (Vessel Traffic Service)	Real-time monitoring and management of maritime traffic	A traffic-monitoring system that provides navigational assistance and information to vessels in specific areas (e.g., ports), helping to prevent collisions and enhance navigation safety
Rete AIS Nazionale (National AIS Network)	Ship identification and position tracking	A nationwide Automatic Identification System (AIS) infrastructure that broadcasts and receives vessel identity, position, course, and speed information, facilitating situational awareness and traffic control
Long Range Identification and Tracking (LRIT)	Global vessel tracking beyond AIS coverage	An international satellite-based system that regularly gathers vessel position data, enabling long-distance monitoring of ship movements for safety, security, and environmental protection

NAVTEX (NAVigational TEXt Messages)	Dissemination of maritime safety information	A system that broadcasts navigational warnings, weather forecasts, and urgent safety messages on medium frequencies, ensuring ships receive timely information on potential hazards
e-Navigation	Integrated digital solutions for maritime communication and safety	A framework that leverages digital infrastructure to streamline maritime data exchange, improve navigational decision-making, and enhance coordination among coastal authorities and vessels
National Maritime Single Window	Centralized reporting for maritime traffic	A unified digital platform mandated by the EU for submitting and managing administrative and logistics data, reducing paperwork and ensuring consistency across all involved authorities

As indicated, none of the technologies currently employed is dedicated solely to detecting individuals at sea in emergency scenarios. This limitation highlights the need to develop a specialized system designed to fill that gap.

1.3 Limitations of current methods

This section outline the current procedures through which the Italian Coast Guard is alerted to emergency situations involving SAR operation and individuals in distress at sea, commonly referred to as man overboard (MOB) incidents. Building on this analysis, the limitations of these methods will be identified, highlighting the need for a new and more effective identification system.

Current procedures for maritime emergencies several critical limitations impede swift rescue. Distress alerts usually rely on conventional channels such as VHF radio calls (often via voice or Digital Selective Calling) and satellite-linked beacons like Emergency Position-Indicating Radio Beacons (EPIRBs) or Personal Locator Beacons (PLBs) transmitting through the COSPAS-SARSAT system [3]. Although these Global Maritime Distress and Safety System measures are essential, they remain fundamentally constrained by human and technological factors. Initial detection of a MOB often depends on human witnesses and manual reporting: if a person falls overboard unnoticed or a lone sailor is involved, no immediate distress signal may reach the Coast Guard, a situation frequently proving fatal in single-handed, night time, or stormy conditions [25]. Even when others do raise an alarm, the process of relaying a mayday by voice is prone to error and delay – the caller may miscommunicate the vessel's position or have their message garbled by poor radio quality, and precious minutes are lost conveying details that an automatic system could transmit instantly. Indeed, traditional radio-based techniques can be hampered by distance, stress, and weather, all of which can make communication difficult and delay the start of rescue operations [3]. Technological safeguards like EPIRBs and PLBs also have inherent limitations: they typically must be activated (automatically or by a person in the water), and a person overboard without such a device or unable to trigger it will derive no benefit from these systems. Moreover, not all vessels carry advanced alerting equipment – large SOLAS-class ships are required to have automated distress transmitters, but the majority of smaller boats are "voluntary" craft not legally mandated to carry radios or beacons [3]. In addition to the previously discussed elements, further challenges arise even after that a distress signal has been received, particularly in the coordination and execution phases of the search operation, which is the topic that we care most. Upon receiving such an alert, the Coast Guard typically deploys both aerial and naval units.

The aircraft is an Agusta Westland 139 (AW139) whose technical specifications make it highly suitable for operations in diverse environmental conditions. The vehicle is equipped with a high-performance electro-optical FLIR camera system, typically mounted on the nose or underside of the aircraft. This stabilized turret includes both daylight and infrared sensors, allowing the crew members to perform visual scans from an elevated position. Once detected, the crew members guide the various naval vessels (Classe 300 Ammiraglio, Classe 800) toward the point of interest. Therefore, the SAR process remains susceptible to limitations. The system's effectiveness still heavily depends on human operators' ability to identify people or vessels that may appear only as a few indistinct pixels within a vast maritime expanse. This reliance on human perception introduces a margin of error, making the operational process both complex and prone to oversight under certain conditions.

This double gap, due to both the difficulty of receiving the initial distress signal and the inherent limitations in accurately identifying the person in distress, presents a significant challenge to timely and effective SAR operations. Previous studies [3][25] proposed solutions aimed to enhancing communication and accelerating the deployment of Coast Guard units. In contrast, this thesis focuses specifically on improving the identification phase, with the goal of increasing the precision and speed of target recognition, thereby optimizing the overall efficiency of SAR missions.

1.4 Research objective

Beyond the development of the model, as outlined in the project description, it is essential to establish specific objectives that ensure its responsiveness and adaptability across various scenarios. In the process of this research, we identified few fundamental characteristics that are critical in maritime emergency situations:

- High Sensitivity to Detail: The model must be capable of detecting even the smallest visual cues. Given the complexity of maritime rescue operations, it is imperative that the system can accurately recognize a human body, even when it is almost fully submerged or in extreme weather conditions.
- Minimization of False Positives: Emergency response protocols involve pro-

cedures that are both time-sensitive and resource-intensive. A high rate of false positives could lead to unnecessary deployments, increasing operational costs and delaying interventions where they are truly needed. However, in life-threatening situations, the system must strike a balance between precision and recall, ensuring that no potential distress signal is overlooked while maintaining operational efficiency.

- Flexibility in handling different situations: although the "man overboard" emergency is the one we are most interested in, SAR operations involve many situations where boats, ships, and cargo are involved. For this reason, we wanted to ensure that the model is also able to be used in situations involving boat-boat and boat-man interactions.

A deeper examination of the second point highlights the critical need to balance false positives and false negatives. On one hand, an excessive number of false positives may trigger unnecessary emergency responses, leading to operational inefficiencies, increased costs, and potential safety risks for rescue personnel deployed in non-critical situations. On the other hand, false negatives pose an even greater concern, as they result in the failure to detect individuals in distress, directly endangering human lives. This trade-off between minimizing unnecessary interventions and ensuring that no life-threatening situation goes unnoticed represents a fundamental challenge in model optimization.

The evaluation of these two factors, along with the methods employed to achieve an optimal balance, will be further explored in Chapter 5 during the model assessment.

1.5 Thesis structure

This thesis includes both research conducted on the subject and the development of the proposed model. More specifically, the structure is as follows:

- Chapter 2 examines current state-of-the-art models for object identification. This section explores the latest advancements in object detection technologies, questioning their potential adaptability to our specific needs. Additionally, it explore models specifically designed for maritime surveillance, evaluating their strengths and limitation. This chapter will result in a deep evaluation of the current models on which we will build our model design.

- Chapter 3 provides a comprehensive overview of the dataset used in this work. It details the procedures adopted for dataset construction, including the acquisition of raw data, the sampling and framing strategies applied, and the criteria used for filtering relevant frames. The chapter also presents an analysis of the data and metadata distribution. In addition, it introduces supplementary open-source datasets incorporated for training and benchmarking purposes.

- Chapter 4 is dedicated to the architectural design of the proposed model, offering a comprehensive breakdown of its core components, functionalities, and underlying methodologies. This chapter delves into the rationale behind the chosen approaches, discussing the selection of algorithms, training phase details, and optimization strategies that contribute to the model's robustness.

- Chapter 5 presents the model evaluation, highlighting its performance, key results, and the challenges encountered during development. The evaluation metrics from the literature will be reviewed and explained. This chapter also discusses the model's explainability, to contextualize its limitations and areas for potential improvement.

- Finally, the concluding chapter provides a summary of the work conducted, reflecting on the key findings while outlining future directions for research and ethical considerations related to the deployment of such technologies.

Chapter 2

State of the Art

An essential initial step in conducting a complex research project such as this is a comprehensive literature review. In this chapter, after a brief introduction to the main computer vision architectures, we will review the most advanced object identification technologies currently developed. We will then explore the main solutions applied specifically to maritime surveillance. The chapter concludes with an evaluation of the studies conducted, along with a critical assessment of the limitations and challenges of these technologies, highlighting the main areas for improvement. This will lead to conclusions about the most suitable architecture on which the development of our model will be based.

The analysis of the state-of-the-art (section 2.2) will focus mainly on the Small Object Detection (SOD). This task, like standard object detection, is situated within the broader domain of computer vision, which includes various methods designed to acquire, process, and interpret visual data.

2.1 Current architectural paradigms

This section presents the key concepts related to architectures that form the basis of most of the solutions that will be discussed in the state-of-the-art section. Although the focus is primarily on networks designed for image analysis, the discussion also briefly addresses CNN-based extensions developed for video processing. These structures extend traditional CNN frameworks to incorporate temporal dynamics, enabling the extraction of spatio-temporal features essential for understanding motion and continuity across frames.

2.1.1 Single-frame oriented architectures

Convolutional Neural Networks (CNNs) are a specialised class of artificial neural networks designed to process data with grid-like topology, such as 2D images. Unlike traditional fully connected networks where every neuron is linked to all activations in the previous layer, CNNs are built to exploit spatial hierarchies in data via convolutional filters, making them highly effective in tasks like image classification and recognition. The architectural structure of a CNN typically consists of the following layers: (1) convolutional layers (feature extraction), (2) activation functions (non-linearity), (3) pooling layers (dimensionality reduction), (4) fully connected layers (inference/classification), and (5) output layer (typically SoftMax or sigmoid for classification).

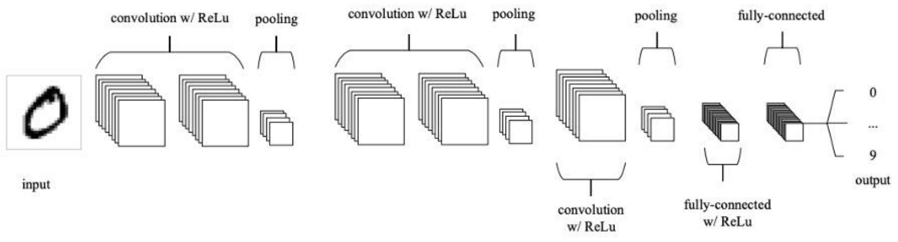


Fig. 2.1: An usual structure of CNN architecture

Figure 2.1 shows a visual example of how these layers are concatenated, specifically applied to one of the first historical visual tasks, the recognition of handwritten digits. The layers of CNNs are iteratively stacked to pyramidally extract information from the input data. In each “stack”, the convolutional and pooling layers work in tandem to extract progressively more abstract and informative features, layer by layer. These features are then passed to the fully connected layers, which act as a compact neural network charged with aggregating and interpreting the extracted information. This hierarchical feature extraction and interpretation process unfolds as follows:

- Early layers focus on detecting low-level features such as edges, corners, and simple textures
- Middle layers capture more complex patterns, including shapes and recurring visual motifs.
- Deeper layers encode high-level semantic information and object-specific representations, enabling tasks such as object recognition or classification.

Figure 2.2 makes the hierarchical structure of CNN models more understandable, showing in particular the VGG-16 model, an evolution of the VGGNet, proposed by Simonyan and Zisserman in 2015 [38].

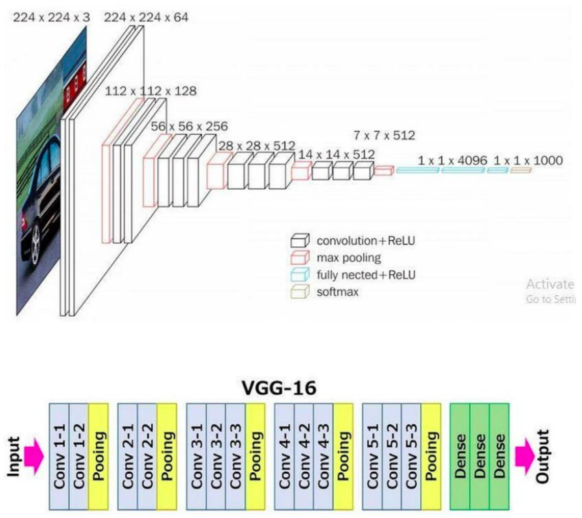


Fig. 2.2: VGG-16 layers structure

Building upon the structure of CNNs, several architectures have been developed to address the object detection task with increasing levels of accuracy and efficiency. These architectures are commonly categorized into two groups: two-stage detectors and single-stage detectors, depending on how the detection pipeline is organized.

Two-stage object detectors, often referred to as region-based methods, are characterized by a sequential pipeline that separates the processes of region proposal and classification. In the initial stage, a Region Proposal Network (RPN) or a similar module identifies a set of candidate regions in the image that are likely to contain objects. These proposed Regions of Interest (ROIs) are then passed to a second stage, where a CNN backbone is used to extract high-level feature representations from each region. These features are subsequently fed into a classification head and a bounding box regressor to assign class labels and refine localization predictions.

In contrast, single-stage object detectors streamline the detection pipeline by eliminating the explicit region proposal phase. These models treat object detection as a unified regression problem, simultaneously predicting object class probabilities and bounding box coordinates directly from densely sampled spatial locations in the feature map. CNNs in this context are responsible for encoding global image features and producing dense predictions over a predefined set of anchor boxes or grid cells. This design allows for significantly faster inference, making single-stage detectors particularly suitable for real-time applications.

More recently, transformers [42] have been applied to object detection to model long-range dependencies and global context. These architectures rely solely on attention mechanisms to model dependencies across input data, allowing them to model global relationships within the input data without relying on locality or sequential structure (as in CNN and RNN). At the core of the Transformer architecture is the self-attention mechanism, which enables each element of the input to dynamically weight its relevance in relation to all others. This global reasoning capability contrasts with the inherently local operations of convolutional layers, offering the advantage of capturing long-range dependencies that are often crucial for detecting spatially distant or contextually ambiguous objects. When applied to visual data, such architectures often receive as input a sequence of embedded image tokens, sometimes augmented with positional encodings to preserve spatial information lost due to the lack of inherent ordering in the attention mechanism. Transformer architectures can eventually be integrated with a convolutional backbone for initial feature extraction. The extracted feature maps are then flattened and passed to the transformer, which processes them to produce contextualized representations. These representations can then be used to predict object locations and categories, avoiding traditional stages such as region proposal generation or anchor box matching.

2.1.2 Video-oriented architectures

When transitioning from images to video data, the foundational principles of object detection remain consistent. However, the task becomes significantly more complex due to the need to process multiple frames rather than a single static image. Employing a single convolutional neural network (CNN) to sequentially analyze individual frames introduces a computational bottleneck, particularly in the context of high-resolution or high-frame-rate video (>30fps[1]), leading to slower recognition performance. Studies such as [36] have explored progressively more sophisticated approaches. These methods basically rely on the static object detection results produced by an image-based object detector, followed by the aggregation of temporal information in a post-processing stage. Thus, the logic behind is the concatenation of information, features, or even detectors through a chain to extend the analysis along several frames. More recent advancements [41] [45] have moved beyond this pipeline by incorporating temporal dynamics directly into the training phase. These approaches match feature maps across sequential frames or predict object proposals over time, allowing the model to learn spatiotemporal patterns more effectively and thereby improving performance on video-based detection tasks.
The computational bottleneck is still evident when employing a CNN-RNN architecture (fig.2.3(a)), which combines 2D-CNNs with a recurrent neural network (RNN) to propagate information across sequential frames. Although this architecture still processes frames individually, the recurrent structure enables the model to capture temporal continuity by recursively integrating information from preceding frames.

[1]Frames per second

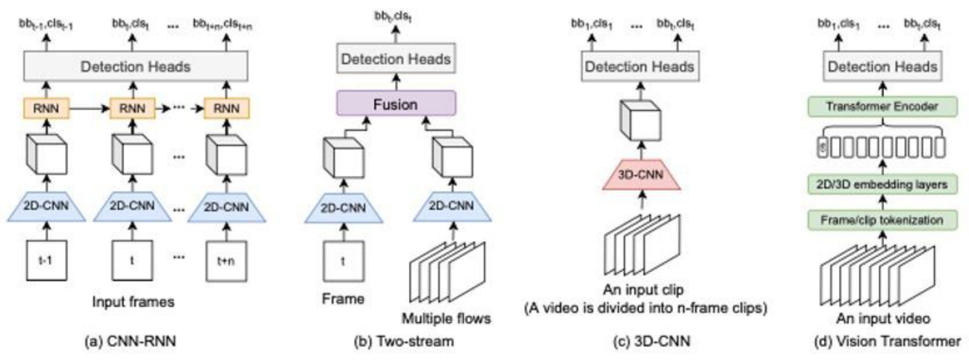


Fig. 2.3: Architectural structures for SOD in videos

While theoretically convenient, it falls short in effectively modeling the temporal logic inherent in dynamic tasks such as action recognition, where distinguishing between temporally similar but semantically distinct actions (e.g., opening versus closing a door) is crucial. A notable variation of this architecture was proposed by Carreira J. and Zisserman A. [9] that replaced the standard RNN with a Long Short-Term Memory (LSTM) network. This modification leverages the LSTM's capacity to encode internal states and model long-range temporal dependencies, thereby offering improved performance in tasks that require an understanding of temporal ordering.

An alternative baseline approach for processing video input is the use of three-dimensional convolutional neural networks (3D-CNNs) (Fig.2.3(c)). These models operate on short video clips, typically divided into sequences of n contiguous frames, enabling the extraction of both spatial and temporal features simultaneously. In particular, the study by Carreira J. and Zisserman A. [9] at Google even proposed the 3D ConvNets combined with LSTM as an evolution of the 3D-ConvNet architectures. The 3D ConvNets apply spatio-temporal convolutional filters that extend across both spatial dimensions and the temporal axis. Unlike 2D CNNs, which process individual frames independently, 3D CNNs capture motion dynamics by directly convolving over successive frames. As the input propagates through the network, the temporal dimension is progressively compressed via strided pooling, allowing the model to construct hierarchical spatio-temporal representations.

Another relevant approach for video-based analysis is the two-stream architecture (fig.2.3(b)), which separates the modeling of spatial and temporal components into two distinct flows. One stream receives a single RGB frame to extract spatial information, while the other takes multiple frames encoded as optical flow to capture motion dynamics, effectively embedding temporal information relative to each frame. Two independent 2D-CNN processes the data that will be fused before the detection head for the final output. In [9], Carreira and Zisserman extended this paradigm by proposing a two-stream model based on the Inception-V1 architecture. In their implementation, one stream processes a sequence of five consecutive RGB frames, while the other handles the corresponding stack of optical flow data— a 5×10 flow representation, consisting of 10 optical flow frames for each of the five intervals.

The previously discussed approaches are typically less prevalent on the object detection task, while are widely used in task like action recognition (on which [9] focus on) or anomaly detection. In contrast, an emerging and increasingly influential direction for object detection in video is the adoption of Video Transformer architectures (fig.2.3(d)). The cornerstone of this category of architectures is the TransVOD [46], which employs multiple video frames as input to spatial transformers, followed by a temporal transformer that further aggregates and refines the learned features. This dual-transformer design enables the model to simultaneously connect object queries and memory encodings across space and time. Building upon this framework, TransVOD++ introduces a Hard Query Mining (HQM) strategy to address the redundancy of object queries—particularly in scenarios with dense scenes or a large number of target. The latest evolution, TransVOT, is designed for real-time video object detection. It integrates two input frames via a correlation module at shallower layers and employs a final tubelet linking module to ensure temporal coherence in object tracking and localization across frames.

2.2 Literature Review on Small Object Detection

As introduced at the beginning of this chapter, our studies have mainly focused on Small Object Detection (SOD), as it is more relevant to the objective of our research. The next sections will define more precisely under what terms, according to the literature, an object is defined as such, highlighting the main limitations encountered. Then, a deep examination on the specific models developed for SOD will be performed, dividing these solutions between one-stage detectors, two-stage detectors, Image Transformers and mixed architecture model.

2.2.1 Definitions and limitations highlighted in literature

The SOD survey by Mahya Nikouei et al. [31] highlights the wide range of definitions in the literature, which change according to the standard adopted and the context of application. In the MS COCO dataset, small objects are typically defined as those occupying a spatial area smaller than 32×32 pixels (equivalent to less than 1,024 square pixels), which correspond to 10% of the total image area in high-resolution optical imagery. Therefore, all evaluation work MS COCOs-based rely on this definition. In satellite and aerial imagery, the challenge of detecting small objects is further exacerbated by their minuscule appearance. For example, moving objects captured in satellite video sequences frequently measure less than 20×20 pixels, emphasizing the difficulty associated with their detection due to their limited size.

Alternative approaches in the literature define small objects based on their relative size within the image, using criteria such as occupying less than 1% of the total image area [48]. This relative threshold is highly sensitive to image resolution and is particularly pertinent in high-resolution imagery, where objects may contain a sufficient number of pixels yet still be considered small due to their proportionally minor presence within the overall scene.

This intrinsic constraint significantly limits the visual information (texture, colour, and shape) available for small objects, thereby hindering the network's ability to perform accurate classification and localization. In addition, it can happen that, in situations where small and large objects coexist, small objects are occluded by larger objects and their extracted features behave as a noise due to their relatively weak values. Besides the evident structural features, another major limitation is the lack of suitable datasets tailored for small object detection. Popular benchmarks like MS COCO or Pascal VOC include only a limited number of small-object instances, often insufficient to train models that generalize well to these targets. The class imbalance caused by the underrepresentation of small objects leads to biased learning and reduced detection accuracy.

These challenges are particularly pronounced in the maritime domain, as highlighted by Rekavandi A. et al [36]. Indeed, they are amplified by a range of environment-specific factors. Visual surveillance in the maritime environment is typically based on aerial or satellite imagery, where even large vessels can appear as small targets due to the distance between camera and object. This also leads to problems related to rapid perspective changes and rapid zoom shifts. Maritime scenes are also characterized by dynamic illumination conditions caused by water reflections, wave movements, and rapidly changing weather. These factors introduce considerable noise and variability in video frames, degrading detection reliability. Furthermore, camera motion resulting from ship swaying or UAV flight instability introduces jitter and blur, making temporal consistency across video frames difficult to achieve. Maritime datasets, in contrast to those for terrestrial or urban settings, are scarce and often lack the diversity and volume required for training robust models.

Conventional object detectors, often designed and trained on datasets dominated by medium and large objects, struggle to generalize to smaller instances. This shortfall is primarily due to aggressive down-sampling operations, such as max pooling or striding, which significantly reduce the spatial resolution of feature maps and thus erase critical details necessary for identifying small objects [31]. Consequently, to achieve reliable detection of small objects, it has been necessary to develop architectural innovations and adopt specialized network designs tailored to address their unique challenges. In this context, the literature has primarily focused on two main detection paradigms: two-stage and one-stage approaches. Two-stage detectors, such as Faster R-CNN, first generate region proposals and then perform classification and refinement, offering higher accuracy. In contrast, one-stage detectors, such as the YOLO and RetinaNet families, perform object localization and classification in a single forward pass, enabling real-time inference with competitive performance. The next sections will explore the state-of-the-art of both paradigms, highlighting their architectural characteristics, strengths, and trade-offs with respect to small object detection in complex environments such as the maritime domain. Both paradigms, two-stage and one-stage, are basically 2D-CNN, while later we will also explore image transformers such as ViT and some mixed architectures such as the DETR family models.

2.2.2 Two-stage detectors

As previously outlined, two-stage object detectors are characterized by a two-step pipeline in which the model first identifies a set of candidate regions likely to contain objects. The first example of this approach was R-CNN (Region-Based CNN)[17], which introduced this paradigm by using selective search to propose about 2000 regions, extracting CNN features for each, and classifying them with SVMs (Support Vector Machine). This yielded high detection accuracy but was extremely slow due to per-region processing and multi-stage training. SPP-Net (Spatial Pyramid Pooling Network) [19] improved R-CNN by sharing convolutional computations. It applies CNN to the whole image once and pools feature maps into fixed-size regions for each proposal, greatly accelerating detection. Fast R-CNN [16] further streamlined training by integrating proposal classification and bounding-box regression into one network and using ROI pooling, achieving 9 times faster training and 213 times faster inference than R-CNN. Fast R-CNN also learned to jointly refine

box coordinates and predict classes, eliminating separate SVMs.
Faster R-CNN [37] introduced a learnable Region Proposal Network (RPN) to replace slow external proposal methods. The RPN shares the backbone CNN features and generates region proposals in real-time, making the pipeline nearly end-to-end. Faster R-CNN achieved breakthrough accuracy with significantly higher speed. However, like earlier two-stage models, its performance on small objects depended on the resolution of feature maps and anchor sizes. R-FCN (Region-Based Fully Convolutional Network) simplified the two-stage pipeline by using position-sensitive score maps instead of per-region fully connected layers. After proposals are generated (as in Faster R-CNN), R-FCN applies a single convolutional pass to classify all proposals, achieving comparable accuracy to Faster R-CNN with less computation. This fully convolutional approach preserved spatial information and reduced computation, but it did not specifically introduce multi-scale features for small objects.

Mask R-CNN [18] extended Faster R-CNN by adding a parallel branch for object mask prediction and introduced ROI Align for more accurate feature pooling. The improved alignment is convenient for small object detection because it maintains pixel-level details in the pooled features, solving the quantization problem of ROI pooling. Although the goal of Mask R-CNN was instance segmentation, its higher localization accuracy proved useful for detecting smaller objects. Feature Pyramid Networks (FPNs) [26] were a key improvement for multiscale detection. FPN augments two-stage detectors with a top-down pyramid of feature maps, combining high-resolution low-level features with strong semantics from higher layers. This provides rich feature representations at multiple scales, dramatically improving detection of small objects that would otherwise be nearly invisible in the top layer of deep networks. For example, Faster R-CNN with FPN became a strong baseline for small objects by detecting on fine-grained feature maps.

Cascade R-CNN [7] addressed the mismatch between the training IoU and test

IoU by employing a series of detectors with increasing IoU thresholds. As proposals are refined through the cascade, the detector becomes more attuned to precise localization. While not designed specifically for tiny objects, Cascade R-CNN improved overall detection quality, which can indirectly benefit small object accuracy by reducing missed detections at high IoUs. Libra R-CNN focused on balanced learning to overcome distortions that hinder the detection of difficult examples (such as very small objects). It introduced three components: IoU-balanced sampling (to ensure training includes sufficient hard-positive examples), a balanced feature pyramid that more smoothly blends multiscale features and a balanced L1 loss to achieve consistent gradient magnitudes. These adjustments yielded more uniform attention across scales and easier versus hard examples, thereby improving detection for small, low-contrast objects. In general, two-stage detectors with multi-scale features (FPN) and balanced training tend to excel in SOD scenarios, often outperforming one-stage methods in recall and precision on tiny objects.

2.2.3 One-stage detectors

The single-stage detection paradigm was first introduced by the YOLO (You Only Look Once) [33] framework, which significantly improved detection speed. The original YOLOv1 [33] divides the image into an $S \times S$ grid and for each cell predicts bounding boxes and class confidences. This design enabled real-time speeds (about 45fps) and simple end-to-end training. However, YOLOv1 struggled with very small objects – each object had to fall wholly within one grid cell, limiting localization precision for tiny targets. It achieved lower recall and accuracy than two-stage methods on smaller objects, in part due to its coarse downsampling and the lack of multi-scale feature maps. Subsequent YOLO variants introduced significant improvements. YOLOv2 (YOLO9000) [34] added anchor boxes (pre-defined bounding box priors) and multi-scale training, allowing the network to detect a wider range of object sizes more effectively. It also upgraded the backbone and applied batch normalization, yielding better accuracy especially on smaller objects than YOLOv1. YOLOv3 [35] further incorporated ideas from FPN: it uses three prediction layers at different scales (feature map strides 32, 16, 8) to detect large, medium, and small objects respectively. This multi-scale design, together with a deeper backbone (Darknet-53) and logistic objectness loss, substantially boosted YOLO's performance on small objects. YOLOv4 [4] continued the trend of improving one-stage detectors with advanced features: Weighted-Residual-Connection (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT), mish-activation, mosaic data augmentation (which aids small-object training by context mixing), and improved loss functions. These later YOLO versions achieved accuracy closer to two-stage models while maintaining high speed. For example, YOLOv4 demonstrated state-of-the-art results in 2020 with 65 fps on GPU, partly by better detecting small instances using enhanced features and training tricks. YOLOv5 [20], developed by Ultralytics, represents one of the first object detection models specifically designed with industrial applications in mind. It adopts a modular architecture that supports extensive customization and facilitates deployment across a wide range of platforms. The model can be exported to multiple formats—including ONNX, CoreML, and TFLite—enabling seamless integration into various hardware environments. A key feature of YOLOv5 is its scalable design, characterized by multiple model variants ("n", "s", "m", "l" and "x"), each offering a different trade-off between computational complexity and detection accuracy. This flexibility makes YOLOv5 particularly well-suited for real-time applications on resource-constrained embedded systems.

Another influential one-stage model is the Single Shot MultiBox Detector (SSD) [29]. SSD also employs multiple feature map scales: it generates default bounding boxes of various sizes and aspect ratios on several layers of the CNN, from fine to gross resolution. Each default box is scored and adjusted to match objects. SSD was one of the first to show that one-stage detectors can be nearly as accurate as two-stage. However, on very small objects SSD's performance lagged. Later research found that SSD's shallow layers still lacked the necessary detail or that training was biased toward medium/large objects. Data augmentation and larger input resolutions (e.g. SSD512 vs. SSD300) were needed to improve small object results. RetinaNet [27] closed the remaining accuracy gap by addressing class imbalance with Focal Loss. RetinaNet uses an FPN backbone (like a two-stage detector's neck) but with a one-stage dense head. The focal loss down-weights easy negatives and focuses learning on hard examples (such as small, hard-to-detect objects). This innovation allowed one-stage detectors to achieve accuracy on par with two-stage models on challenging datasets. RetinaNet's strong performance on small objects is attributed to the combination of multi-scale feature pyramids and the training bias correction from focal loss, which together improve the classifier's sensitivity to subtle small-object features.

Several one-stage detectors have been proposed to improve SOD by eliminating dependence on predefined anchors, introducing so-called anchor-free detectors. CornerNet [23] pioneered this by detecting each object via its top-left and bottom-right corner keypoints rather than regressing from anchors. It uses a heatmap to locate likely corner positions and an embedding to group pairs of corners, eliminating anchor boxes entirely. This approach can localize objects of any scale without tuned anchors, which might benefit small objects that do not align well with preset anchor sizes. CenterNet [13] built on CornerNet by predicting an object's center in addition to corners (a keypoint triplet) to improve pairing accuracy. By ensuring corners are paired only if a center keypoint exists between them, CenterNet reduces false matches, which is helpful when small objects are close together or numerous.

The trend of anchor-free models has also been embraced by the YOLO family, which introduced this advancement from YOLOv8 onward, developed again by Ultralytics. Released in 2023, YOLOv8 [21] marked a pivotal evolution in the YOLO series by streamlining the architecture and enhancing performance, particularly in detecting small and overlapping objects. Building on the foundation laid by YOLOv5, it offered a more flexible and unified framework capable of supporting various computer vision tasks, including object detection, classification, and segmentation. Improvements in model design (refined loss functions, optimized convolutional structures, and task-specific heads) contributed to significant gains in both accuracy and efficiency. Subsequent versions continued this trajectory of innovation. YOLOv10 [2] introduced architectural refinements aimed at further boosting performance in resource-constrained environments. Key advancements included a decoupled head architecture, separating classification and regression tasks to reduce interference, and task-aware downsampling, which improves representational capacity without increasing computational load. Enhanced multi-scale feature extraction modules, such as improved Spatial Pyramid Pooling (SPP), further strengthened the model's robustness across varied object sizes and scenes. The latest release, YOLOv11 [22], advanced the series with a focus on minimizing inference latency while maintaining accuracy. It eliminates the traditional Non-Maximum Suppression (NMS) step by adopting NMS-free training and inference, enabling end-to-end predictions that inherently avoid duplicate detections. In addition, YOLOv11 incorporates advanced architectural blocks and lightweight attention mechanisms that reduce computational overhead while maintaining high accuracy.

The great flexibility and modularity of the YOLO family of models leaves also room for the development of numerous custom implementations, which have sought to adapt and improve YOLO models specifically for a particular domain of application. One example that particularly worth mentioning for this thesis was presented by Goran Oreski [32], who proposed YOLO*C, a family of YOLO-based models enhanced with contextual information. The approach was applied to YOLOv5 (still an anchor-based model), yielding the corresponding variants YOLOv5C. The architecture preserves the canonical backbone–neck–head pipeline of YOLO, but introduces a novel Multi ConTeXt (MCTX) module, which processes multiscale feature maps and outputs global context classifications such as weather, time of day, and scene type. Several custom implementations are noteworthy, some of which will be examined in detail in the sections on hybrid models. Among them, the YOLO*C approach is particularly relevant, as it will play a central role in shaping the methodological

choices and experimental design of the following chapters.
Overall, one-stage detectors have evolved considerably, addressing many of their early limitations in small object detection. Advances in model architecture have substantially improved their ability to localize and classify smaller objects. One of the key advantages of the one-stage approach lies in its computational efficiency: by directly predicting object classes and bounding boxes in a single pass, these models enable significantly faster inference, making them particularly well-suited for real-time applications and resource-constrained environments such as mobile or embedded systems. However, despite these improvements, one-stage detectors may still exhibit reduced accuracy when detecting extremely small objects, especially in cluttered or low-resolution scenes. This limitation stems from the dense prediction paradigm, which can struggle to extract meaningful features from very small regions due to limited spatial information. Nevertheless, the trade-off between speed and precision continues to favor one-stage models in scenarios where low latency is critical.

2.2.4 Vision Transformer

Vision Transformer (ViT) [12] was the first demonstration that an image can be processed as a sequence of patch embeddings with self-attention instead of convolutional layers. Although ViT was initially proposed for image classification, its principles can be applied to detection tasks: a transform-based detector can theoretically deal with an entire image and resolve features of small objects in the context of the scene. In practice, the application of ViT to sensing has posed challenges to SOD. The ViT divides the image into patches (e.g., 16x16 pixels), so a very small object might occupy only part of a single patch and thus its features are lost. However, the global receptive field of transformers is an advantage: even a tiny object can be recognized if contextual cues (surrounding patterns) are captured by self-attention.

Researchers tested this idea with YOLOS (You Only Look at One Sequence) [15], which adapts a pre-trained ViT for object detection with minimal modifications.

YOLOS treats detection as a sequence prediction problem, inputting image patches and outputting a set of object bounding boxes and labels using the transformer encoder-decoder architecture. Notably, YOLOS does not incorporate an explicit feature pyramid or multi-scale processing – it relies on the transformer's global attention over one scale of patches. The result was a proof-of-concept that a pure transformer (with no CNN or region proposal stage) can perform object detection competitively. However, YOLOS also exposed limitations for SOD. Lacking multi-scale feature maps, YOLOS struggled with the smallest objects, as the fixed patch size and single-scale pipeline failed to capture fine details. The authors noted that YOLOS "was unable to benefit from multi-scale features and achieved limited performance" on smaller instances. Thus, although transformers can theoretically aggregate global context helpful for detecting small objects in cluttered scenes, they require architectural changes to handle scale variability. Recent vision transformers like Swin Transformer incorporate hierarchical feature maps (pyramidal layers) to address this, enabling strong performance on detection including small objects, but those go beyond the scope of plain ViT/YOLOS.

In summary, ViT-based detectors introduce a new paradigm where attention mechanisms can capture relationships that might aid in SOD (e.g. a small boat's presence might be inferred by attending to a wake or the horizon context). YOLOS demonstrated the feasibility of transformer-only detection, but also highlighted that without multi-scale feature fusion, even powerful global attention struggles with tiny objects. The SOD field has yet to fully exploit transformers' potential.

2.2.5 Hybrid CNN-Transformer Models

Mixed architectures combine convolutional networks with transformer modules, aiming to get the best of both worlds. A prime example is DETR (DEtection TRransformer) by Carion et al [8]. DETR uses a CNN backbone to extract feature maps, then passes those into a transformer encoder-decoder that outputs a set of object predictions directly, using bipartite matching to align predictions with ground truth.

This end-to-end approach eliminated the need for hand-crafted processes like non-maximum suppression and anchor box design. DETR achieved accuracy equal to that of previous state-of-the-art detectors on large objects and introduced a simpler pipeline. However, DETR had two notable issues affecting SOD: (1) Slow training convergence – it required very long training schedules to learn the matching, partly because the transformer had to learn spatial relations from scratch. (2) Difficulty with small objects – the original DETR did not use feature pyramids or multi-scale feature fusion, so the transformer operated on a single resolution feature map (usually a 16x downsampling from the CNN). Small objects, which contribute weak signals on a coarse feature map, were often missed or poorly localized. In COCO experiments, DETR's performance on the small object subset was significantly lower than FPN-based detectors.

Researchers quickly developed DETR derivatives to address these limitations. One key improvement was multi-scale feature encoding. For instance, Deformable DETR [47] introduced a deformable attention module that samples a sparse set of points around reference positions on multiple feature scales. By attending only to relevant locations at multiple resolutions, Deformable DETR reduced computation and allowed the model to focus on fine details of small objects. It achieved a $10\times$ faster training convergence than DETR and significantly improved average precision on small objects. Another tweak was adding iterative bounding box refinement (a two-stage variant of Deformable DETR), which boosted accuracy for small objects by letting the network adjust its initial predictions.

Beyond deformable attention, other DETR variants incorporated ideas from traditional detectors. Some works introduced feature pyramid inputs to the transformer or additional high-resolution CNN branches. For example, the FP-DETR [43] used an FPN backbone and focused pre-training of the encoder, enabling the detector to leverage multi-scale features for SOD. This hybrid strategy combines the proven multiscale representation of the FPN with the prediction of the transformer set, giving better results for small objects. Another notable extension is UP-DETR (Unsupervised Pre-training DETR) [11], which pre-trains the DETR model on a proxy task (randomly cropping and reconstructing image patches) to give the transformer some spatial “knowledge” before fine-tuning for detection. While not a direct architectural change, this pre-training helped DETR learn positional embeddings that improved its detection of small objects and accelerated training convergence.

Similarly, DAB-DETR [28], Conditional DETR [30], and others guided the transformer’s queries with explicit positional priors (akin to anchors) to make learning easier – these methods have been shown to improve DETR’s precision on small objects by ensuring the model doesn’t have to discover from scratch where in the image small items might appear. Moreover, a recent specialized variant called SOF-DETR (Small Object Focused DETR) [14] explicitly targeted SOD: it injected a bias towards small-size feature maps and normalized the attention to concentrate on finer details. By feeding the transformer multi-scale representations of the input, SOF-DETR was reported to better capture tiny object features and achieved superior results on small object benchmarks.
In essence, mixed CNN-transformer detectors like DETR and its derivatives are promising for SOD because they can capture complex relationships (e.g. a small object amidst background clutter) through attention, while recent innovations ensure that high-resolution features are not lost. These architectures are computationally heavy, but they point toward a future where explicit proposals or anchors may be unnecessary even for small objects, as the model itself learns where and how to attend to tiny regions of interest.

2.3 SOD applications in the maritime domain

As already explained extensively, maritime environment presents a challenging application for small object detection. Targets such as distant ships, boats, buoys, or even persons overboard typically appear as small, low-resolution blobs in video frames. Moreover, the ocean environment introduces additional difficulties: sunlight reflections on water and wave clutter cause rapid illumination changes, which can fool detectors with false highlights. Adverse weather (fog, rain) and the dynamic sea state reduce visibility, making small objects even harder to distinguish from background. Aerial surveillance (from drones or planes) often views objects at oblique angles, causing ships to appear in various distorted shapes and scales within the scene. Camera platforms on unstable aircraft add jitter and motion blur. All these factors mean that generic object detectors need careful adaptation for maritime SOD.

One of the best-known example of a one-stage model applied to the maritime environment is ImYOLOv3 [10], an improved YOLOv3 adapted to maritime SOD. This approach incorporated spatial and channel attention modules into YOLOv3 to help the model distinguish true ships from sea background. This approach demonstrates that one-stage detectors can be adapted to the sea domain by embedding mechanisms to ignore irrelevant regions (like water) and emphasize object-specific features.

Recent advances in transformer-based detection models offer further potential but are not yet widely adopted. DETR-based models, for instance, could leverage global context to decide whether a small blob is a boat or just foam: the transformer can encode the surrounding water pattern, horizon line, and other ships in view to inform its detection. This contextual awareness is particularly useful at sea, where an object's meaning can depend on context (a cluster of pixels is likely a distant ship if it lies on the horizon and not moving like waves). A leading example is Ship-DETR proposed by [44] which incorporates high-low frequency (HiLo) attention into its intra-scale feature module to better extract both edge-level and global features. Usually, transformers applied in this context are used to struggle with the scale variation: ship features of different scales represent a pain point for these models. To address this issue, Wang Y. et al. incorporated the bidirectional feature pyramid network (BiFPN) to optimize cross-scale feature fusion to further enhance

the representation of small-scale ship features.
What has been said above demonstrates that the major object detection frameworks (two-stage, one-stage, and transformer-based architectures) can all be applied to small object detection in maritime contexts, but each has trade-offs. The next section will highlight the results of the studies performed on the state of the art, analyzing in detail the strengths and weaknesses of the main architectures examined in order to identify the most suitable approach for our application case.

2.4 Summary of findings and research direction

In situations where human lives are at risk, the key is to find an optimal compromise between accuracy and speed of calculation. Being generally very accurate, two-stage detectors are essential in maritime safety applications. However, they lose out in terms of speed and computational load. For real-time monitoring, the slower inference of two-stage models can be a limitation. By contrast, single-stage detectors are preferred for their speed and, with domain-specific improvements, can achieve satisfactory accuracy for real-time surveillance. Transformed and mixed architectures also undoubtedly represent an excellent opportunity, but their technological maturity is still immature.

That said, the state-of-the-art seems to confirm that the YOLO family of models is one of the most cutting-edge, thanks in large part to Ultralytics' continued development of new implementations, which has led to significant gains in detection accuracy, model efficiency, and ease of deployment across various platforms. These characteristics have positioned YOLO as an hard-to-beat benchmark.

However, as this thesis will demonstrate, the direct application of a predefined model to maritime data often yields suboptimal results due to the domain-specific challenges of the marine environment. More critically, the literature lacks applications of such models specifically for human identification, which is fundamentally different from the more commonly addressed problem of ship or general object detection. This gap motivates the development of a model tailored for human identification, as explicitly requested by the Coast Guard. However, as already discussed in the objectives of this thesis, the proposed solution will not only be completely vertical to detect humans in maritime environments, but also be robust to effectively handle scenarios involving various types of vessel or human-vessel interactions.

In this context, this thesis introduces a research idea that aligns with the hybrid architectures presented in Section 2.2.4 and is inspired by the YOLO*C model proposed by Goran Oreski [32], previously discussed in Section 2.2.2. Building on these foundations, Chapter 4 will present a YOLO-based implementation extended with an attention module, thereby combining the efficiency of one-stage detection with an improved ability to focus on the most relevant visual information.

Chapter 3

Dataset

This section provides a review of the datasets used in this study, explaining their sources, their generation, acquisition, preprocessing and transformation processes, as well as the reasons behind their selection as resources for research. Two datasets are examined: MOBDrone [6] and CoastGuard. The first is an open-source dataset developed and made publicly available by the Artificial Intelligence for Media and Humanities (AIMH) research group of the CNR. The second comes from clips provided directly from the Italian Coast Guard for our exclusive use and forms the basis for the main tasks addressed in this research.

The two datasets, MOBDrone and the Coast Guard (CG) dataset, were employed in a series of training trials aimed at exploring their potential both individually and in combination. Rather than adopting a fixed sequential fine-tuning strategy, multiple experimental configurations were tested to assess how each dataset contributes to model adaptation in the maritime domain. The MOBDrone dataset, being larger and cleaner, was initially considered a strong candidate for providing general maritime contextualization, while the CG dataset was expected to inject domain specificity with its direct focus on human and vessel detection in SAR scenarios.

This experimental design was motivated by several factors. First, the CG dataset on its own is too limited in scale to support robust training, making it necessary to leverage additional data sources. Second, evidence will prove that transferring from general-purpose pretrained models, such as those trained on COCO, to a highly vertical task often yields suboptimal results, reinforcing the need for a domain-aligned dataset such as MOBDrone. However, the results of the exploration of alternative training strategies will be presented and critically analyzed in the experimental section of this thesis (sec.5.2).

3.1 MOB Dataset

The MOBDrone dataset is a large-scale aerial image collection specifically designed to support the development and evaluation of automatic visual detection systems for man-overboard scenarios. This dataset was introduced by Cafarelli et al. [6] as part of a broader research initiative conducted by the Artificial Intelligence for Media and Humanities (AIMH) Laboratory of the Institute of Information Science and Technologies (ISTI), part of the Italian National Research Council (CNR). The development of the MOBDrone dataset forms part of a broader interdisciplinary research program at AIMH focused on the intersection of computer vision, machine learning, and digital humanities, targeting drone-based Search and Rescue (SAR) in marine environments.

3.1.1 Dataset Composition

The MOBDrone dataset includes 126.170 frames, extracted from 66 full high-definition (FHD) video sequences, each recorded by a DJI Phantom 4 Pro V2 drone operating at altitudes ranging from 10 to 60 meters. The original videos were captured in 4K resolution and subsequently downsampled to 1080p to facilitate more computationally manageable frame-level analysis. These videos represent a controlled marine environment featuring simulated rescue situations with people at sea and boats and natural distracting elements such as buoys and floating debris.

Classes	Person	Boat	Surfboard	Wood	Life-buoy
IDs	0	1	2	3	4

Table 3.1: Class IDs - MOBDrone

Each frame of the dataset was annotated by researchers with bounding boxes using the CVAT (Computer Vision Annotation Tool) platform. A total of 181,689 bounding boxes have been manually annotated, of which 113,408 correspond to the principal class of interest: person (overboard). The remaining annotations are distributed across four additional categories: boat ($\approx$ 40.000 instances), wooden debris ($\approx$ 16.000), lifebuoy ($\approx$ 10.000), and surfboard ($\approx$ 2.000). The inclusion of these categories introduces critical distraction and ensures that detection systems are trained not merely to detect generic objects but to discriminate the target class under ambiguous visual conditions. The categories are encoded with a unique class IDs as highlighted in Tab. 3.1).

3.1.2 Data Format and Split

The dataset is distributed in both image and video formats, along with COCO-style JSON annotation files supporting multiple levels of granularity. Three variants of the annotation set are provided:

- annotations-5-custom-classes.json: includes all five categories;
- annotations-3-coco-classes.json: restricts labels to those shared with the COCO dataset (person, boat, surfboard);
- annotations-person-coco-classes.json: focuses solely on the ”person” class.

Being in a COCO format with a ”.json” extension, the labels required a transformation process so that they would comply with the format required by the YOLO family models. YOLO models require a specific directory structure in which the training, validation, and test sets are organized into separate folders, each containing the images and the labels posed in different sub-folders maintaining identical filenames but differing in file extensions (commonly, jpg or png for images and txt for the labels). The annotation files follows a strict format, where each row is 5 element long: class id + bounding box coordinates expressed in the format (x_center, y_center, width, height), all normalized to the [0,1] range with respect to the image dimensions (e.g. [0, 0.1, 0.3, 0.01, 0.04]). The resulting folder structure is illustrated in Fig.3.1 using a hierarchical tree representation.

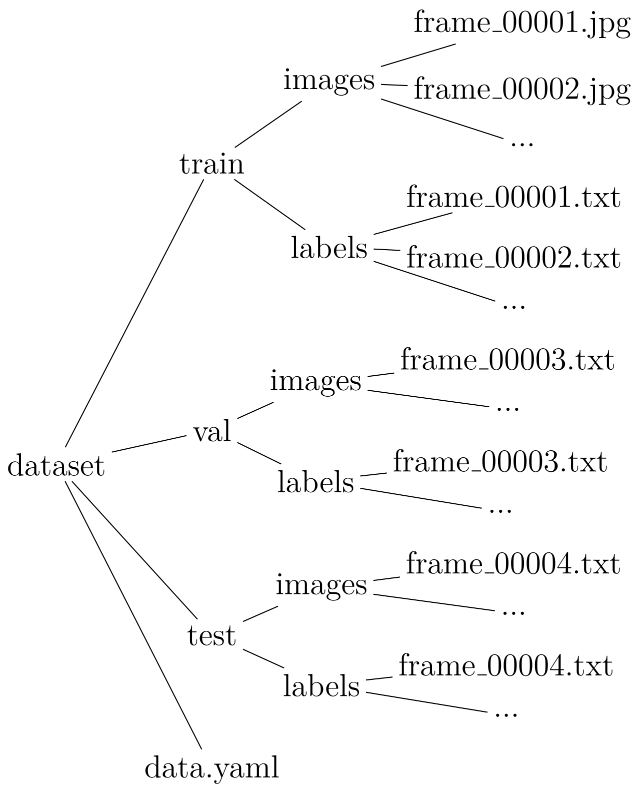


Fig. 3.1: Dataset folder structure required by YOLO

To ensure consistency and reproducibility of the experimental evaluation, the authors of the dataset recommend a default data split that separates training and test sets based on video file prefixes. Specifically, all images with file names beginning with “DJI_0915,” totaling 88,568 frames, are intended for training, while images beginning with “DJI_0804,” totaling 37,604 frames, are reserved for testing. This subdivision is intended to minimize scene overlap by preventing frames from the same (and therefore likely similar) video from being in the same subset and to avoid data leakage between training set and test/val set. In this study, the recommended split was adopted to align with the authors' benchmarking protocol and facilitate a fair comparison with existing results.

However, the recommended subdivision has two criticisms. First, it does not provide the validation split, but it is easy to overcome this problem. The second, more important, emerged during an exploratory data analysis, where it was found that the authors' recommended subdivision would be wrong and lead to poor training performance. The reason is that there is an imbalance in the distribution of classes. Some classes present in the test set (e.g., life jacket and surfboard) seem not to appear in the training set. As a result, the model has no way to learn these categories and inevitably does not recognize them. To overcome these problems, starting from the recommended division, we rebalanced the distribution of classes and derived a validation set partly from the training set and partly from the test set, taking care to place frames from the same video in the same division to avoid data leakage.

3.1.3 Scene Diversity and Acquisition Protocol

Notably, approximately 27.7% of the dataset frames contain no labeled object, thereby offering an informative background distribution for false positive analysis and background modeling. This aspect distinguishes MOBDrone from other aerial datasets, which often overrepresent object-containing frames, leading to performance inflation under real-world operational constraints.

To enhance robustness and generalizability, the dataset was designed to capture a varety of acquisition context. These include variations in altitude, illumination, camera angles, and subject posture. Frames are evenly distributed across six discrete altitude bands from 10 m to 60 m, with the largest frame counts recorded at 30 m, 40 m, and 50 m.

The scenes were primarily filmed using a perpendicular view (90° angle), although selected clips employ a 45° oblique perspective, enabling comparative performance testing across camera geometries. The video sequences depict two trained divers, one male and one female, performing a range of overboard states, including upright swimming, prone floating, and unconscious postures.

Despite this effort in generalise the scene of acquisition in terms of altitude, viewpoint, and subject posture, the data deeply lack variety in the environmental and atmospheric condition. All video sequences were captured over a calm, flat sea surface, under favorable weather and lighting conditions, thereby excluding scenarios involving high waves, rain, fog, low visibility, or strong winds. This factor is a serious limitation for the real-world applicability of a solution developed solely from these data.

3.1.4 Data analysis and Benchmarks

To gain a deeper understanding of the label distribution and spatial characteristics of the annotated objects in the MOBDrone dataset, an exploratory data analysis (EDA) was conducted on the labels. This analysis aim to assess the class imbalance, bounding box positioning, and size distribution across the dataset, with the goal of identifying potential biases or structural patterns that could influence model training and detection performance.

Figure 3.2 showcases the analysis performed on the training set, and it summarizes the distribution of annotated instances per object class along with heatmaps of their spatial and dimensional distributions. The bar plot (fig. 3.2(a)) reveals a class dominance by the ”person” category in the training set (approx. 80,000 instances), followed by ”boat” and ”wood”. The ”surfboard” and ”lifebuoy” classes are notably underrepresented, which may result in suboptimal generalization for these minority categories. However, this is due to the nature of the dataset. In fact, the validation set and test set proportionally reflect these measures, giving them representativeness compared to the training set.

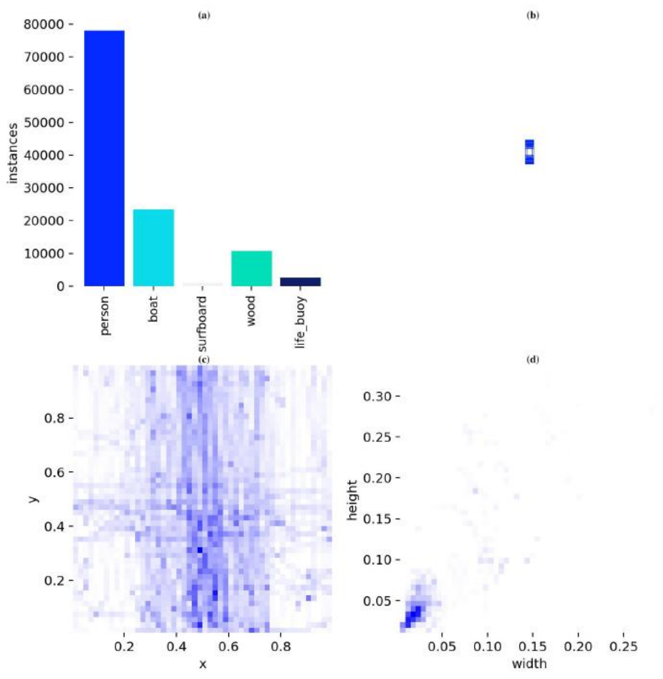


Fig. 3.2: Label Distribution and Spatial Density - MOBDrone Dataset

Actually, the dominance of the “person” class is exactly what we were looking for when we started looking for new data to expand instances of the Coast Guard dataset. In fact, as we will see in the next section, our main dataset has a clear underrepresentation of this category, which is instead our main detection target. On the other hand, the CG dataset is rich in frames containing vessels. Therefore, the choice of this dataset stems from the need to re-balance the total amount of instances of our main classes (person and boat).

Fig.3.2(b) shows the overlap of the size and shape of the bounding boxes. As we can see, the bounding boxes all tend to have a very similar and very narrow shape. This reflects a dataset dominated by objects with a high and narrow aspect ratio, which is particularly characteristic of upright or semi-submerged human figures captured from an aerial viewpoint.

The two heatmaps at the bottom (fig.3.2(c) and (d)) show important insights about the positions and the sizes of the bounding boxes. The left one represent the 2D density of the bounding box centroids across the image plane in normalized coordinates $(x, y) \in [0, 1]^2$. The distribution is highly concentrated along the vertical center and middle-third region of the frame, reflecting the typical positioning of objects when captured from a 90° aerial viewpoint. The right one illustrates the joint distribution of bounding box width and height. Most bounding boxes are clustered in the lower-left region of the plot, meaning that the majority of objects occupying less than 10% of the frame area, confirming that is about Small Object Detection (as explained in sec. 2.2.1). This is consistent with the small physical size of humans and floating objects as captured from medium to high drone altitudes.

The pairplot in fig.3.3 give a deeper analysis about the spatial and dimensional relationships of the bounding boxes, expanding the previous fig.3.2(c) and adding also univariate histograms along the diagonal about the frequency of the bounding boxes attributes (x, y, w, d). This expanded view reveals several important patterns.

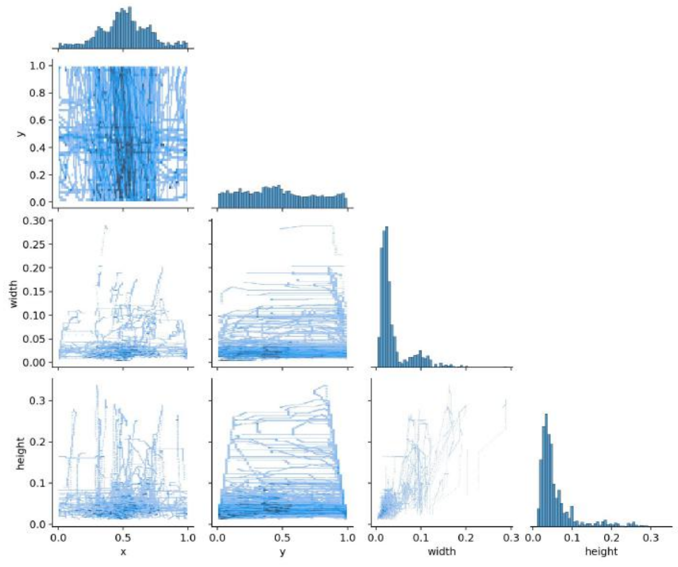


Fig. 3.3: Pairwise Relationships Between Bounding Box Features - MOBDrone Dataset

The distributions of the center coordinates indicate a broad horizontal spread and a more centered vertical tendency.The width and height distributions, by contrast, are heavily skewed toward small values, reinforcing the observation that most annotated objects occupy a minimal portion of the image frame. The upper triangle, particularly the width–height cell, is rendered as a 2D histogram that visualizes the joint frequency of bounding box shapes. The resulting vertical concentration highlights the predominance of narrow and vertically elongated bounding boxes.

The paper published by the authors of the MOBDrone dataset [5] provides not only a detailed account of the dataset's development and structure but also presents a comprehensive benchmark evaluation of state-of-the-art object detection models. In this thesis, we report a subset of those results, focusing on the main performance metrics of models that were also reviewed in the state-of-the-art section. The benchmark includes nine object detection architectures, grouped into three categories: (i) anchor-based CNNs (e.g., Faster R-CNN, YOLOv3, TOOD, VarifocalNet), (ii) anchor-free CNNs (e.g., CenterNet), and (iii) Transformer-based methods (e.g., DETR, Deformable DETR). All models were pre-trained on the MS COCO dataset, and their performance was evaluated on MOBDrone without fine-tuning, using Average Precision (AP) at multiple IoU thresholds.

Model	**AP@50**	**mAP@.50:.95**
VarifocalNet	0.378	0.144
TOOD	0.314	0.116
Deformable DETR	0.199	0.075
Faster R-CNN	0.126	0.041
CenterNet	0.124	0.041
DETR	0.128	0.040
Mask R-CNN	0.109	0.033
YOLOv3*	0.011	0.009

Table 3.2: Comparison of selected object detectors evaluated on the MOBDrone dataset

As reported in Table 3.2, VarifocalNet achieved the highest detection performance with an AP@50 of 0.378 and mAP@[.50:.95] of 0.144, followed by TOOD and Deformable DETR. However, all models showed relatively modest performance, underscoring the challenges of detecting people at sea from aerial viewpoints, particularly when trained on datasets from terrestrial environments.

A class-wise evaluation shows that even the best models often correctly localize person instances but misclassify them as belonging to irrelevant COCO categories such as ”bird” or ”kite”. In particular, VarifocalNet detected approximately 81.8% of all person instances, but only 28.5% were correctly classified. This suggests that the primary source of error is misclassification rather than localization, reinforcing the need for domain-specific fine-tuning. These results emphasize that off-the-shelf detectors trained on general-purpose datasets like COCO are not directly transferable to the challenging maritime search-and-rescue context of MOBDrone. Fine-tuning on in-domain data, combined with class balancing and shape-aware data augmentation, is likely necessary to achieve operational performance.

3.2 Coast Guard Dataset

The Coast Guard dataset is derived from a comprehensive framing, filtering and pre-processing process applied to a collection of aerial videos captured during real search and rescue (SAR) operations. These videos offer a invaluable resource for research in this area, both for their verticality on the topic and truthfulness of situations captured. Unlike simulated scenarios, the videos document real operations conducted in 2024. This authenticity becomes crucial for the model’s ability to learn and the transferability of that learning to the real world scenario.

3.2.1 Dataset Composition

The CG dataset consists of 7.906 frames, extracted from a total of 206 videos in 1920x1080 resolution quality with a total duration of about 229 minutes recorded at

40 fps. Although the original clips contain about 342.957 frames, a filtering process was applied to ensure high-quality data and maximize the diversity of frames. This selection was designed to improve the generalization capability of the model by providing a representative and diverse set of visual inputs. Being a detection task, the model does not need consecutiveness of frames, and it is therefore important to make the model learn to generalize as much as possible with ever-changing frames, avoiding overfitting on often similar situations. As already said, the videos represent SAR operations, therefore the objects depicted are mainly persons and boats. In addition, it has been made a distinction between person on the boat and person overboard.

Classes	Person	Boat	Person Overboard
IDs	0	1	2

Table 3.3: Class IDs - CG

3.2.2 Framing, filtering and annotation processes

Given the high frame rate of 40 fps, a direct frame-by-frame extraction would have introduced considerable redundancy, which was already present in the previous dataset. To this end, after the framing phase, a multistage filtering process was conducted to improve the overall quality and relevance of the selected frames.

The first stage of the filtering process focused on the color properties of each frame, analyzed in the Hue-Saturation-Brightness (HSB) color space. This color representation, unlike RGB, allows us to trace the average color distribution of pixels to identify frames that are particularly predominant in a particular color. Each frame was evaluated based on the distribution of brightness and saturation values. Excessively dark or overexposed frames were removed, as they are commonly attributed to ”total black” or ”total white” frames. These typically occur due to changes in zoom or transitions between visor modes.

In a second phase, each frame was subjected to a visual quality assessment, with the aim of eliminating technically poor images. Four quality metrics were calculated for each frame: sharpness, contrast, clarity, and resolution. Once calculated, the quality metrics were weighted and summed into a single quality score, which was then normalized across the entire dataset to ensure comparability. The weight of each metric was assigned based on a statistical method that looks how much each metric varied within the dataset, assigning higher weights to metrics with less variability based on the standard deviation.

A threshold was then applied to discard all frames that did not meet the minimum acceptable quality level. Additionally, studying the distribution of the frame quality index, it was noted that some outliers had extremely high quality values even though they were very ”dirty.” These were mostly extremely ”pixelated” frames that eluded certain metrics and ”escaped” filtering. Therefore, a maximum threshold was also set to ensure compliance with plausible situations. This step was crucial for filtering out a large portion of the blurry frames caused by motion, camera instability, or loss of focus, factors that can significantly hinder the accuracy of manual annotation and model training. A percentage of these ”low-quality” frames were, however, retained to increase the model’s generalization ability.

Finally, a last filtering stage has been performed to avoid the visual and informative redundancy between consecutive frames. This step was designed to ensure that the dataset included only frames that were meaningfully distinct from one another. Three similarity metrics were calculated:

- Color histogram distance in HSV space, capturing global color distribution differences;
- Structural Similarity Index (SSIM), measuring local structural and perceptual changes;
- Template matching on grayscale.

The frames were discarded if at least two out of three measures resulted in ”positive” similarity to the last saved frame (not necessarily the previous one). At the conclusion of the filtering process, only frames that met the criteria for brightness, quality, and dissimilarity were retained. This multi-stage refinement reduced the original set of approximately 343,000 frames to a curated subset of 8,112 frames, stored for annotation.

The annotation process was also composed by two steps. In the initial phase, a YOLOv11 model pretrained on the COCO dataset was used to generate preliminary bounding box via inference. These raw predictions were stored and later used to accelerate the manual labeling process. Given the domain gap between the COCO dataset and the specific visual context of maritime SAR operations, the model's predictions were often imprecise or incomplete, and many frames lacked any predicted labels. The second phase involved the use Label Studio [40], an open-source data annotation platform. The preliminary labels served as a rough starting point which was then manually reviewed, corrected, and completed within the software. Although the automatic inference significantly reduced the initial workload by pre-annotating some frames, the manual refinement process remained labor-intensive and required a huge amount of hours.

3.2.3 Data Format and Split

Label Studio offers great flexibility in exporting annotations, supporting a wide range of formats, including YOLO. This compatibility simplified the alignment of annotation output with the required data structure, described in Section [?], facilitating integration into the training pipeline.

Once annotation was complete, the dataset was split into training, validation, and testing sets using a 70-20-10 ratio. The splitting process was designed to preserve class distribution across the subsets, ensuring that each class, especially less frequent ones such as ”person overboard,” was proportionally represented across all splits. Additionally, to avoid data loss and artificially inflated performance, frames from the same source video were assigned to the same subset. This constraint was essential to maintain the independence of the validation and testing sets, ensuring a reliable assessment of the model's ability to generalize to unseen data.

3.2.4 Data analysis

The analysis in this sections mirrors the procedure applied to the MOBDrone dataset (Sec.3.1.4) and aims to reveal class imbalances, spatial biases, and size characteristics of the annotated objects, with a deeper study of the bounding boxes and labels relations.

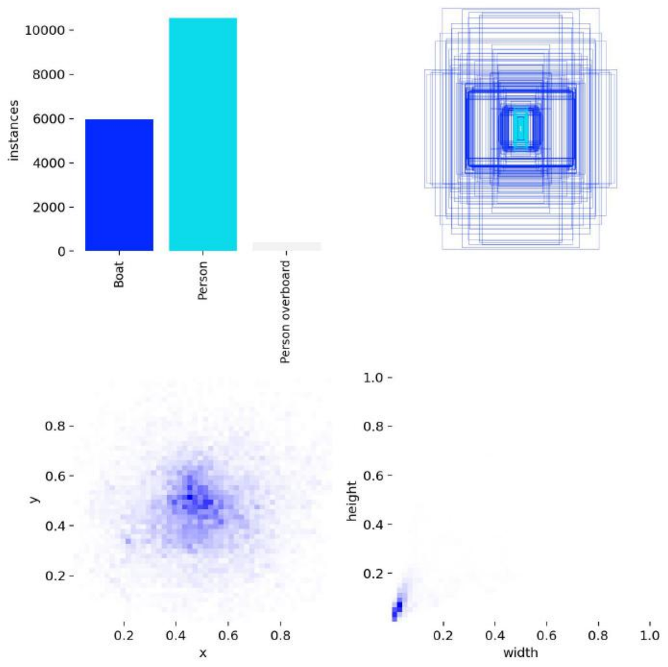


Fig. 3.4: Label Distribution and Spatial Density - Coast Guard Dataset

The bar chart in Fig.3.4(a) shows a significant dominance of the Person and Boat class, followed by very sparse occurrences of Person overboard. This sharp imbalance contrasts with the MOBDrone dataset (see Fig.3.2(a)), where the Person Overboard category is predominant. However, these values require careful contextualization. A closer examination of the dataset reveals that it is strongly boat-dominant. The apparent abundance of human instances is largely explained by frames in which several individuals are present simultaneously on the same vessel, rather than by a balanced distribution of people across different maritime scenarios. A more in-depth quantitative analysis, focusing on the distribution of classes and their relationships, allows us to identify three recurring configurations in the context of search and rescue (SAR) operations: (i) frames containing only one or more vessels, typically associated with patrolling, inspection, or protection of cargo or merchant ships, potentially threatened by piracy; (ii) frames featuring vessels with people on board, often related to migrants or illegal fishing operations; (iii) frames featuring people at sea, sometimes accompanied by a small vessel in the immediate vicinity, situations that require particular operational attention. In quantitative terms, 67,5% of the frames include a single vessel, 28,4% contain more than one vessel, while only 4,1% of the frames contain no vessels at all, representing cases of people at sea or simple background images.

Specifically, 5.335 frames featuring a single boat, of which 1.754 (32,88%) also include at least one person. There are 2.245 frames with multiple boats, and 614 of these (27.35%) show potential interactions by also including at least one person. A total of 450 frames depict a person in the sea, and in the majority of these (73,56%, or 331 frames), a boat is also present, highlighting rescue scenarios. Additionally, 199 frames contain no annotated objects.

Among the frames containing people, most contain only a small number of individuals, typically between 1 and 7, while frames with larger groups (more than 10 people) are rare. Some ”outliers” show up to 30 or even 46 individuals in a single frame, corresponding to crowded rescue scenes or larger vessels. Overall, the dataset is dominated by sparse human presence, with occasional high-density annotations.

Moving forward with the analysis, the heatmap in Fig.3.4(c) reveals that bounding box centroids (x, y) cluster tightly around the center of the image, exhibiting a higher concentration than in the MOBDrone dataset. This could introduce a location prior during training, which should be contrasted using data augmentation or bias-aware modeling techniques. This graph slightly differ from the one of the previous dataset, in which the bounding boxes where placed in the central part but over the entire longitude and latitude of the image.

The bounding box overlays (Fig. 3.4(b)) indicate a strong diversity in the bounding boxes size and shape. Differently from the previous dataset, some objects can even takes the entire image, while others just a very small portion of pixels. The reason behind is that certain videos refer to cargo inspections or close supervision of boats to verify the number of men on board. However, the joint distribution of width and height (Fig.3.4(d)) reveals that most bounding boxes occupy very small regions of the frame, confirming that this dataset also falls within the domain of Small Object Detection. The spread in height appears limited, with an even more compact cluster than observed in MOBDrone, reflecting tighter homogeneity in object size.

A deeper analysis regarding the bounding box area, normalized to the frame size, reveals a sharp contrast between object categories. Vessels occupy the largest average area (0.095), while humans on board and unidentified objects are significantly smaller, with mean values of approximately 0.0035–0.0038. In particular, individuals at sea are represented by tiny bounding boxes with an average relative area of just 0.000145 pixels, approximately 10 pixels, posing significant challenges for reliable detection. On average, vessels are approximately 31 times larger than people, highlighting the disproportionate scale and inherent difficulty in detecting small, vulnerable targets in open-ocean environments.

Figure 3.4 shows the pairwise relationships among bounding box attributes. As in the MOBDrone analysis, diagonal histograms show heavy skew toward small widths and heights, while scatter plots indicate minor correlations between spatial and size dimensions.

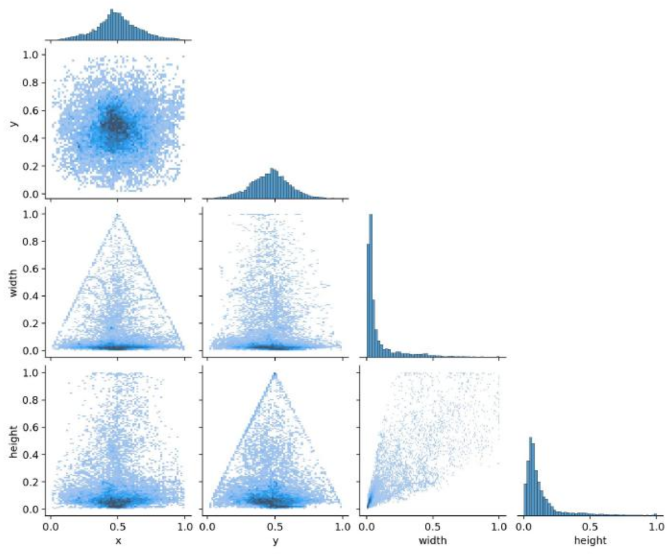


Fig. 3.5: Pairwise Relationships Between Bounding Box Features — Coast Guard Dataset

The histograms on the diagonal of the graphic matrix in Fig.3.5 confirm what was said previously: the bounding boxes occupy a very small portion of space and are concentrated in the central area of the image. Overall, the graph confirms that, unlike the MOBDrone dataset in which the subjects were deliberately distributed across the image in a simulated environment, in real situations the situation is different, since the operator must try to keep the subject in the center of the image.

In summary, while the CoastGuard dataset contributes a diverse and boat-heavy annotation set with greater variability in object sizes and shapes, the MOBDrone dataset provides a complementary, person-heavy set with highly centered and consistently sized bounding boxes. Together, these datasets form a coherent training corpus for robust multi-class maritime object detection. To further clarify their main similarities and differences, table3.4 provides a side-by-side comparison of the two datasets, highlighting key aspects such as size, object classes and environmental conditions.

Feature	**MOBDrone**	**Coast Guard (CG)**
Frames	126,170 (from 66 videos)	7,906 (from 206 videos, filtered from 343,000 original frames)
Resolution	4K (3840×2160) captured, downsampled to 1920×1080	Full HD (1920×1080)
Classes	Person, Boat, Surfboard, Wood, Life-buoy (5)	Person, Boat, Person Overboard (3)
Class distribution	Person heavily dominant; minor classes underrepresented	Boat and Person on board dominant; Person overboard very rare
Environment	Calm sea, good light; no weather variability	Wide variability: day/night, RGB/IR, different sea states, visibility levels
Viewpoint	Mostly nadir (90°), some oblique (45°)	Operational views, often centered manually on targets
Scenarios	Two divers simulating overboard postures	Real SAR: patrol, inspections, migrants, illegal fishing, rescues
Context data	None (conditions constant)	Sensor (RGB/IR), time of day, sea state, visibility

Table 3.4: Comparison between MOBDrone and Coast Guard (CG) datasets

3.2.5 Context data

The model developed in this research project (which we will explore in detail in chapter 4) requires the integration of additional data, called context data. This is generic and, indeed, contextual information about the entire image, not about a particular detail.

Context information is contained in four variables, stored in a json dictionary. Following the YOLO philosophy, each frame has its own json file containing the context data, named exactly like the frame but with a different extension. They are stored in a different subfolder, but at the same level as the train/val/test folders. The following contextual attributes were selected:

- **Image Channels (*RGB or IR*)**: This variable indicates whether the frame was captured using a standard RGB camera or an infrared (IR) sensor. Including this information helps the model learn modality-specific visual patterns, particularly important in the Coast Guard contexts where both imaging systems are frequently used interchangeably. Certain objects may exhibit different thermal signatures in IR imagery compared to their appearance in RGB. This distinction is essential, as identical classes can appear significantly different depending on the imaging modality used during the mission.
- **Datetime (*Mattino, Pomeriggio, Notte*)**: This variable captures the time of day at which the frame was recorded. Lighting conditions vary dramatically across these temporal intervals, affecting object contrast, shadowing, and overall visibility. Including this information allows the model to learn how detection performance may be influenced by ambient light levels, and to adjust its confidence or interpretation accordingly.
- **Sea State (*Calmo, Moderato, Mosso, Molto Mosso*)**: This variable describes the surface conditions of the sea. Under turbulent conditions, wave activity and water reflections can introduce visual noise, increasing the likelihood of false positives. Furthermore, boats may appear tilted, submerged, or distorted due to wave dynamics. By incorporating sea state as a feature, the model can potentially learn to account for these variations and adjust its bounding box predictions or class confidence based on environmental difficulty.
- **Visibility (*Scarsa, Bassa, Media, Nitida*)**: This variable refers to the clarity with which objects can be visually identified in a frame. It is influenced by environmental factors such as fog, lighting, and atmospheric haze,

but in training phase has been considered also the technical limitations like sensor resolution or motion blur, since they simulate low-visibility conditions. This decision was made to simulate the impact of poor visibility on detection performance and to help the model learn how to adjust its confidence when object perception is compromised.

These contextual variables are not intended as primary detection inputs but serve as auxiliary signals that enhance the model's capacity to interpret object presence under varying operational conditions. The inclusion of these variables allows the model to acquire a broader semantic awareness of the visual context, making it easier to discriminate between useful cues and visual noise.

Contextual data is not treated as independent signals, but rather as integral components of the model's internal representation. As we will explore in the next chapter, the model employs an attention mechanism that enables the joint encoding of contextual and primary input features within a shared latent space. Through this process, contextual information is not processed alone, but is fused to capture and exploit their interdependencies and how they influence each other. By embedding the contextual signals within the latent space of visual features, the model is able to capture and exploit their mutual dependencies. This representation facilitates more robust and adaptive learning, as it allows the model to dynamically adapt its internal processing based on the contextual relationships embedded in the data.

To obtain these data starting solely from the images we utilized the Gemma-3 large language model developed by Google [39]. In order to maximize the extraction of information from each frame, we exploited the on-screen metadata displayed on the helicopter camera. Indeed, the input that the model received was composed by three images: the full frame, a cropped region from the top-left corner, and another from the top-center of the image. In those position the frame contains, respectively, the time and the sensor used (IR or RGB). By explicitly providing these cropped regions, the model was able to directly read this contextual information from the screen, rather than guess it from visual content. Regarding environmental conditions such as visibility and sea state, the model was guided to describe each frame based on predefined parameters specified within the prompt. Through prompt engineering, we steered the model to respond by filling a specific output layout. Then we extracted the log selecting the answers that matched a specific pattern of characters and we saved the results in json files.

While this approach proved effective, it also presented certain limitations. The 4-bit quantized version of Gemma-3, chosen as a compromise between computational feasibility and model capacity given the large volume of data, inevitably reduced performance compared to higher-precision variants. Moreover, in cases of uncertainty, the model often defaulted toward median values of the variables, leading to biased distributions and requiring subsequent manual refinement of the extracted metadata. These factors highlight both the promise and the challenges of applying LLMs for systematic metadata extraction in large-scale visual datasets.

In addition, the integration of contextual data was applied exclusively to the Coast Guard dataset due to its extensive environmental and situational diversity, which provides a wide range of contextual information. In contrast, as discussed in Section 3.1.3, the MOBDrone dataset lacks such environmental variability. Consequently, the contextual data it provides is highly repetitive and does not contribute meaningful variation across different scenarios.

Chapter 4

Model architecture: MaYOLOc

This section presents the model developed during the research and development period for the task introduced in previous chapters. Its name, **MaYOLOc**, encompasses all its main features: **Ma**ritime (the application domain), **YOLO** (the model on which it is based), **C**ontext (the developed implementation).

The choice to rely on a YOLO architecture has already been extensively explained in the chapter on the state of the art. The model's increasing accuracy and optimization of detection speed make YOLO one of the best models when operating in real-time. Furthermore, its modularity allows for the addition of layers and thus the possibility of developing a custom implementation starting from a solid foundation.

Therefore, the first section will provide a detailed description of the YOLOv8 [21] architecture, which is used to illustrate the general functioning of the YOLO family of models. While YOLOv8 serves as the reference example, the implementation remains flexible and, as will be shown later, can be readily adapted to more recent versions such as YOLOv11. Subsequently, space will be given to the developed implementation and the training environment on which the tests were carried out.

4.1 YOLOv8 architecture

YOLOv8 has been released 2023 by Ultralytics, builds upon the YOLOv5 framework with significant architectural innovations, notably an updated backbone and an anchor-free, decoupled detection head. This anchor-free design simplifies the model and eliminates the need for manual anchor tuning, while the decoupled head separates classification and localization tasks for improved specialization.

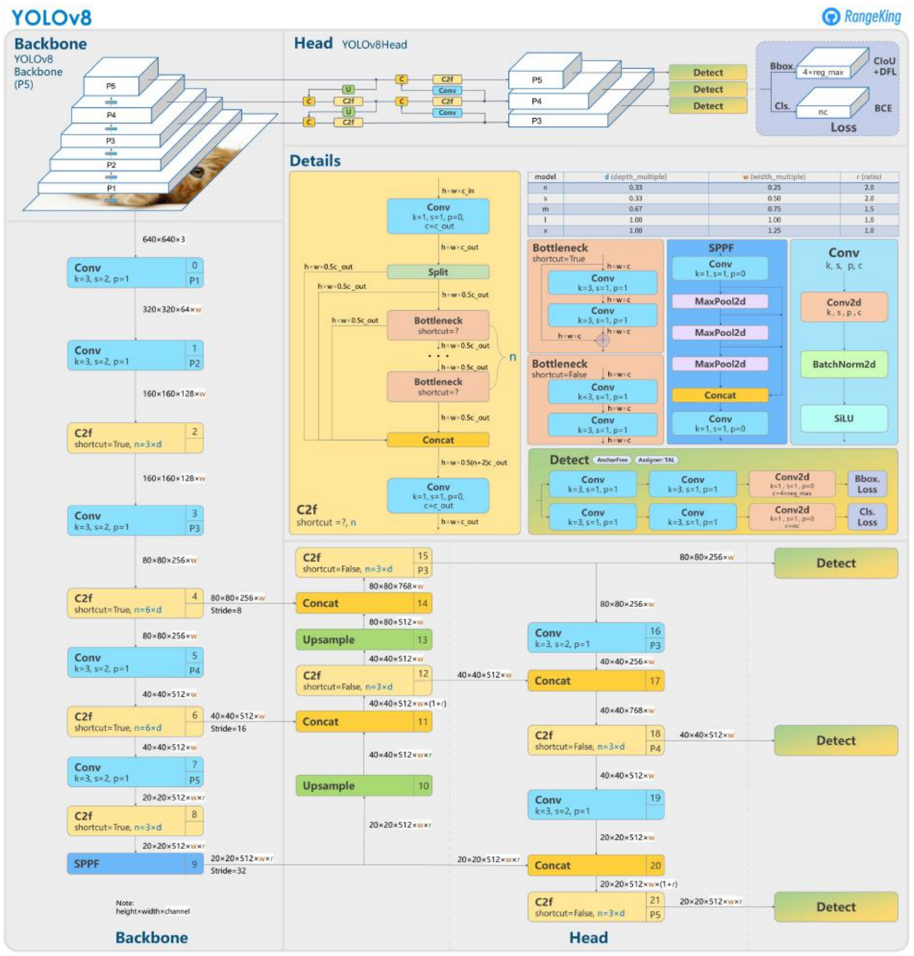


Fig. 4.1: YOLOv8 architecture

The figure 4.1 show in detail every component of the model. The architecture is composed by three components: a backbone for feature extraction, a neck for multi-scale feature fusion, and a head for final object detection.

4.1.1 The Backbone

The backbone is a deep convolutional network (derived from CSP-Darknet53) that transforms the input image into a hierarchy of feature maps at multiple scales. Starting from the input (by default resized to a $640 \times 640 \times 3$ image), a series of convolutional layers with stride 2 progressively downsample the spatial resolution while increasing the channel depth. Each convolution performs a linear operation defined as:

$$y_{i,j,c_{\text{out}}} = \sum_{c_{\text{in}}=1}^{C_{\text{in}}} \sum_{u=1}^{k_h} \sum_{v=1}^{k_w} W^{(u,v)}_{c_{\text{out}},c_{\text{in}}} \, x_{i+u-1,j+v-1,c_{\text{in}}} + b_{c_{\text{out}}} \,, \tag{4.1}$$

where $x_{,i,j,c_{\text{in}}}$ and $y_{,i,j,c_{\text{out}}}$ are input and output feature values, and $W^{(u,v)}$ and b are the learned kernel weights (of size $k_h \times k_w$) and bias. Each convolution in YOLOv8 is followed by batch normalization and a SiLU activation $\sigma(x) = x \cdot \frac{1}{1+e^{-x}}$ (also known as Swish), ensuring stable and non-linear feature transformations. The backbone uses multiple stages of convolution and pooling to produce feature maps P_3, P_4, P_5 at different scales (for example, 80×80, 40×40, 20×20 for an input of 640^2). These maps capture both fine details and coarse semantic information crucial for detecting objects of various sizes.

C2f Module and Bottlenecks

A key component in YOLOv8's backbone is the C2f module, which replaces the earlier CSP C3 block from YOLOv5. C2f stands for Cross Stage Partial with two convolutions. The `details` section in the Fig. 4.1 shows how it is composed internally: it utilizes a bottleneck residual block with two conv layers and a shortcut connection (inspired by ResNet). Formally, if X is the input feature map, the bottleneck computes $F(X) = \text{Conv}_2\Big(\sigma(\text{Conv}_1(X))\Big)$, and with shortcut enabled, the output becomes $X + F(X)$ (identity skip-addition) to ease gradient flow. In the C2f block, the input X is first split along the channel dimension into two parts $[X^1, X^2]$. The second part X^2 passes through n sequential bottleneck units (with n determined by the network depth), producing a transformed output $Y^2 = \mathcal{F}(X^2)$. This is then concatenated with the unmodified X^1:

$$Y_{\mathrm{c2f}} = \mathrm{Concat}\left(X^1, Y^2\right) \tag{4.2}$$

By combining a portion of features that bypass the bottlenecks with the processed features, the C2f module preserves gradient flow and feature diversity. Each bottleneck within C2f uses SiLU activations and maintains the number of channels (expansion factor = 1, so no reduction in channel width across the bottleneck).

Spatial Pyramid Pooling-Fast (SPPF)

At the end of the backbone, YOLOv8 employs a Spatial Pyramid Pooling – Fast module to further enrich the highest-level features with multi-scale context. The SPPF layer performs multiple max-pooling operations of different receptive fields and concatenates them, which mimics multi-scale pooling but in an optimized manner. Concretely, given an input feature map X (e.g. P_5), SPPF applies three sequential 5×5 max-pool operations (with stride 1) to generate pooled outputs Y_1, Y_2, Y_3 of progressively larger effective kernel sizes. If $Y_0 = X$ for notation, this can be written as:

$$Y_1 = \mathrm{MaxPool}_{5\times5}\left(Y_0\right), \quad Y_2 = \mathrm{MaxPool}_{5\times5}\left(Y_1\right), \quad Y_3 = \mathrm{MaxPool}_{5\times5}\left(Y_2\right). \tag{4.3}$$

These pooled features Y_1, Y_2, Y_3 are then concatenated with the original X: $Z = \mathrm{Concat}(X, Y_1, Y_2, Y_3)$, and passed through a final conv layer to produce the SPPF output, which feeds into the neck.

4.1.2 The Neck and Feature Fusion

YOLOv8's neck uses a PAN-FPN style structure (path aggregation network) to combine features from different backbone stages, allowing detection at multiple scales. Features from P_5, P_4, P_3 are first upsampled and concatenated with lower-level features (lateral connections), then passed through additional C2f modules to blend the information. For example, the P_5 output (after SPPF) is upsampled and concatenated with P_4 feature map (from an earlier backbone stage), followed by a C2f block to produce a refined P_4 feature for detection. This top-down path is analogous to an FPN, ensuring higher-resolution features (useful for small object details) are enriched with context from deeper layers. YOLOv8 also includes a bottom-up path (PANet) where lower-resolution features (rich in semantics) are propagated downward (for instance, the refined P_3 is downsampled and fused back with P_4, etc). The C2f modules in the neck replace traditional convolutional merges, preserving the cross-stage partial connections during feature fusion. Overall, the neck outputs a set of merged feature maps at three scales (corresponding to P_3, P_4, P_5 in the diagram) which are fed to the detection head.

4.1.3 Decoupled Detection Head

The YOLOv8 head is decoupled into separate classification and regression subnetworks. For each of the three feature map scales, two parallel conv branches predict class probabilities and bounding box coordinates. In practice, as illustrated in the `detection` section of fig. 4.1, each detection head consists of two Conv layers with BN and activation dedicated to classification, and another parallel conv block dedicated to regression. Let F_k denote the feature map (from the neck) at scale k (where $k \in P_3, P_4, P_5$, corresponding to detecting small, medium, large objects). The classification branch produces a tensor of shape $H_k \times W_k \times C$, where $H_k \times W_k$ is the spatial size and C is the number of classes. Each spatial location (i, j) thus yields class confidences $p_c^{(i,j)}{}_{c=1}^{C}$ via a sigmoid activation (or equivalently a softmax over classes). In parallel, the regression branch produces an output of shape $H_k \times W_k \times 4$ (or $H_k \times W_k \times 4'$ if additional parameters like an objectness score or distribution bins are included). Each location (i, j) in this branch predicts a bounding box $\mathbf{b}^{(i,j)} = (b_x, b_y, b_w, b_h)$, parameterized as offsets from that grid cell. YOLOv8 is anchor-free, meaning (b_x, b_y) might represent the relative center coordinates within the cell and (b_w, b_h) the width and height (often normalized to the entire image) without reference to anchor templates. Notably, YOLOv8 has removed the explicit objectness confidence output that earlier YOLO versions used, instead relying on the class confidence (or an implicit centerness measure) to indicate if an object is present at a location. During inference, predictions from the three scales are combined and filtered (e.g. using Non-Maximum Suppression) to produce the final detected boxes and their class labels. The decoupling of class and box predictions allows each sub-network to specialize, yielding higher accuracy: the classification head focuses on recognizing object categories, while the regression head fine-tunes localization.

4.1.4 YOLO Loss Function

Training for a detection tasks usually involves optimizing a multi-task loss $\mathcal{L} = \mathcal{L}_{\text{box}} + \mathcal{L}_{\text{cls}}$ (summing over all predicted boxes). In addition, YOLOv8 adopts the Distribution Focal Loss (DFL) to further refine localization accuracy. Therefore, YOLOv8's total loss is defined as

$$\mathcal{L} = \mathcal{L}_{\text{box}}^{OBJ+DFL} + \mathcal{L}_{\text{cls}} \tag{4.4}$$

where the localization loss include both the box loss and DFL. Let's delve into each loss in detail.

Bounding box regression loss and classification loss

The bounding box regression loss $\mathcal{L}_{\text{box}}$ in YOLOv8 is based on the Complete IoU (CIoU) loss. For a predicted box b and ground-truth box b_{gt}, the IoU is $\text{IoU} = \frac{|b \cap b_{gt}|}{|b \cup b_{gt}|}$. CIoU extends this by penalizing center distance and aspect ratio mismatch.

$$\text{CIoU}\left(b, b_{gt}\right) = \text{IoU} - \frac{\Delta_c^2\left(b, b_{gt}\right)}{c^2} - \alpha v \tag{4.5}$$

where Δ_c is the Euclidean distance between box centers, c is the diagonal length of the smallest enclosing box covering b and b_{gt}, v measures the difference in aspect ratio, and α is a weight factor.

The classification loss $\mathcal{L}_{\text{cls}}$ in YOLOv8 is a binary cross-entropy loss applied per class (treating the problem as multi-label classification for each predicted box). For a given predicted class probability p_c and ground-truth label $y_c \in 0, 1$ (where only the true class has $y_c = 1$), the loss is $-\Big[y_c \log p_c + (1 - y_c)\log(1 - p_c)\Big]$. This is equivalent to a multi-class cross-entropy since each box is associated with a single ground-truth class.

DFL: Distributed Focal Loss

The DFL [24] requires a dedicated module implemented as a neural network component in the YOLO architecture. It serves a dual purpose: it transforms predictions from classification outputs into precise coordinate values and serves as part of the bounding box prediction refinement process. Indeed, during training the DFL contributes to the calculation of the object loss.

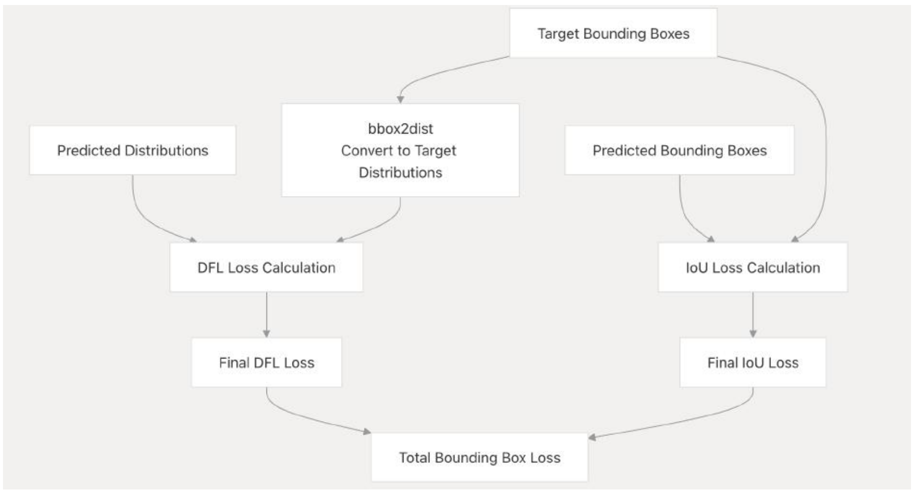


Fig. 4.2: DFL integration in object loss computation - YOLOv8

Its role and its integration within the object loss computation can be highlighted in fig. 4.2. Basically, DFL then applies a focal-loss variant that encourages the predicted distribution to concentrate around the ground-truth value, improving fine-grained localization (especially for small objects).

4.2 Context Integration Through Self-Attention Module: MaYOLOc

In this section, we present the methodology used to integrate contextual information into the YOLO architecture. Incorporating context required both extending and modifying the original architecture. Several key components of the YOLO pipeline were selectively adapted to effectively exploit context-aware features.

First, the dataloader and dataset pipeline were enhanced to include contextual data alongside the standard image-label pairs. Additionally, the loss computation module was reformulated to consider the contributions of context-based predictions. Finally, the model's output layers (the model head) were redesigned to integrate a HybridContext module and the self-attention mechanism that fuses visual and contextual features. These changes allow the network to leverage contextual information throughout the detection process.

The strength of this implementation lies in its incredible flexibility and adaptability across all YOLO architectures from v8 onwards (as long as it follows the archor-free logic). This is possible thanks to the hooking logic developed in the model, which extracts the detection head modules and dynamically hooks the attention module.

4.2.1 Data pipeline

To enable the integration of auxiliary contextual information within the YOLO training framework, the data pipeline was significantly extended. These modifications focus primarily on two critical components: the dataset class and the dataloader construction routine. The standard pipeline of YOLO, which expects only image-label pairs following the YOLO format, was extended to support context annotations, structured as metadata files accompanying each image.

The dataset pipeline begins with the definition of a custom dataset class, *MaY-OLOcDataset*, which inherits from *YOLODataset* within the Ultralytics framework. This subclassing allows for the reuse of standard processing and loading procedures from Ultralytics. The dataset class accepts a parameter *context_root*, which is read from the extended yaml file from which YOLO is used to set the configuration about the dataset, and optionally a *context_mappings* dictionary to encode categorical variables. These context files are assumed to be stored in a structured JSON format and are parsed upon dataset instantiation.

The dataset class also overrides the standard method responsible for loading individual training samples. In addition to retrieving the image tensor and corresponding label targets, this method now also appends a context feature vector. This context vector is derived by indexing the structured metadata associated with each image and converting it into a fixed-length tensor. This vector is appended to the returned dictionary under a designated key, ensuring that both visual and contextual information are simultaneously available during training and validation.

To ensure consistency with the contextual data of the image, augmentation has been disabled. In fact, variations in pixel brightness or color and image mosaics, for example, would not have maintained the consistency of the information. This will certainly be a downgrade in terms of model generalization, but it opens up the development of new augmentation techniques that are consistent with this method. The function to ”transform” the images has been overwritten to perform only the standard configurations such as image resizing and padding.

To support the downstream propagation of these extended data structures, the data loader construction process was adapted. A new routine, *build_maritime_dataset*, replaces the default YOLO data loader creation. This function instantiates the modified dataset class and prepares it for consumption by PyTorch’s DataLoader.

4.2.2 Extended Loss Function

As we will see more in detail in the subsection, the model actually perform a multi-training, since it simultaneously learns to perform detection and predicts the context variables. Although context prediction is not the core focus of the proposed system, it ensures a structurally integrated fallback mechanism, useful in scenarios where explicit context annotations may be missing or incomplete during inference.

Therefore, the loss function is formulated as a weighted sum of two terms: the original detection loss computed from the YOLO head, and an additional context classification loss derived from auxiliary prediction heads. Formally, the total loss is expressed as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{det}} + \lambda \cdot \mathcal{L}_{\text{context}} \tag{4.6}$$

where $\mathcal{L}_{\text{det}}$ denotes the detection loss (bounding box regression, class prediction and DFL discussed in sec. 4.1.4), $\mathcal{L}_{\text{context}}$ is the aggregated cross-entropy loss over all contextual variables, and λ is a tunable hyperparameter controlling the contribution of the auxiliary task. Once again, since the main goal is to improve detection performance through externally provided contextual information, the weight λ is intentionally kept low at this early stage of development. This ensures that the model focuses primarily on detection, while context prediction plays a supporting role. However, any future exploration of context information prediction will involve a gradual increase in λ to better leverage the auxiliary task. For now, context prediction mainly serves as a fallback mechanism to improve robustness in the absence of explicit contextual signals.

Even if the context does not directly shape the detection loss, the influence of context emerges from the way the model's parameters are updated: since the attention layers are conditioned on context and lie within the path that leads to detection outputs, the gradients from the detection loss naturally propagate through them. This means that the model learns to pay attention to features differently depending on the contextual input. In practice, the context shapes the intermediate representations used by the detection head, thereby altering the box regression and classification outputs. As a result, the context steers the model to make detection decisions under varying conditions, indirectly shaping the loss.

This approach ensures that the context-aware mechanism exerts a measurable influence on the detection component while preserving the integrity and comparability of standard evaluation metrics. The impact of contextual integration will be further demonstrated through empirical analysis and visualizations presented in the explainability section (Sec.5.3).

4.2.3 Model architecture: Hybrid Context Module with self-attention

Along with the data pipeline and loss function, the MaYOLOc architecture enhances the YOLO framework with a hybrid context-aware cross-attention mechanism. This module allows the detector to encode scene context either injected externally or inferred directly from the image, and to condition multi-scale features before the detection head. The result is a detection system capable of adapting feature representations according to the operating conditions.

MaYOLOc wraps an Ultralytics YOLO model instantiated from a YAML configuration, retaining the original backbone/neck and the standard Detect head. During initialization a lightweight hooking strategy captures the last three high-level neck outputs (typically $P3, P4, P5$) by registering forward hooks on the final on the last three $C2f/SPPF$ modules. A dummy forward pass is used to infer the channel dimensions $\{C_3, C_4, C_5\}$ of these feature maps. The original Detect head is preserved for inference/training on the fused features (i.e., the head remains structurally unmodified, preserving the loss structure and output format).

Context encoding

Let the hooked multi-scale features be $F_s \in \mathbb{R}^{B\times C_s\times H_s\times W_s}$ for $s \in \{3,4,5\}$. The HybridMCTX module proceeds as follows:

1. **Shared image summary for context prediction.** Each F_s is compressed through a 1×1 projection to a common hidden size h (with $h = 128$), batch-normalized and ReLU-activated:

   $$\tilde{F}_s = \mathrm{ReLU}\Big(\mathrm{BN}\Big(W_s^{(1\times1)} * F_s\Big)\Big) \in \mathbb{R}^{B\times h\times H_s\times W_s}.$$

   Global average pooling yields a per-scale vector $z_s \in \mathbb{R}^{B\times h}$. Concatenating across scales and passing through a shared MLP produces an image-level descriptor:

   $$z = \Big\|_s z_s \in \mathbb{R}^{B\times 3h}, \quad u = \phi(z) \in \mathbb{R}^{B\times 128},$$

   where ϕ is a two-layer MLP with dropout, serving as a compact image-derived context proxy.

2. **Per-variable categorical predictions.** For each context variable v with K_v classes (e.g. `sea_state`:4, `visibility`:4, `datetime`:3, `image_channel`:2), a linear head produces logits $\ell_v \in \mathbb{R}^{B\times K_v}$ from u, and softmax yields $p_v = \mathrm{softmax}(\ell_v)$. These posteriors are treated as predicted context distributions.

3. **Hybrid combination with external context.** At runtime, a dictionary of external labels $\{y_v\}$ may be provided; a sentinel value denotes "unknown." For each v,

   $$c_v = \begin{cases} \mathrm{one\_hot}(y_v) & \text{if } y_v \text{ is valid,} \\ p_v & \text{otherwise,} \end{cases}$$

   applied element-wise so that mini-batches can mix known and unknown entries. The final context vector is

   $$c = \Big\|_v c_v \in \mathbb{R}^{B\times \sum_v K_v}.$$

4. **Low-dimensional context embedding.** A projector maps c into a compact embedding:

$$e = W_2\,\sigma(W_1 c) \in \mathbb{R}^{B\times 64},$$

with $\sigma = \mathrm{ReLU}$. This vector e serves as the query for cross-attention.

Context-aware cross-attention fusion

For each scale s, MaYOLOc constructs keys and values from the spatial features, and uses the global context embedding as a single query:

- **Key/value projections.** Each feature map is projected via learned 1×1 convolutions to obtain d-channel maps (here $d = 64$), flattened spatially:

$$K_s \in \mathbb{R}^{B\times(H_sW_s)\times d}, \quad V_s \in \mathbb{R}^{B\times(H_sW_s)\times d}.$$

- **Query from context.** The embedding e is projected to $q \in \mathbb{R}^{B\times d}$, then reshaped to $B \times 1 \times d$.

- **Scaled dot-product attention.**

$$\alpha_s = \mathrm{softmax}\left(\frac{qK_s^\top}{\sqrt{d}}\right) \in \mathbb{R}^{B\times 1\times(H_sW_s)}.$$

- **Context summary at scale s.**

$$m_s = \alpha_s V_s \in \mathbb{R}^{B\times 1\times d}, \quad M_s = \mathrm{broadcast}(m_s) \in \mathbb{R}^{B\times d\times H_s\times W_s}.$$

The broadcasted summary M_s is concatenated with the original F_s:

$$\hat{F}_s = \mathrm{concat}(F_s, M_s) \in \mathbb{R}^{B\times(C_s+d)\times H_s\times W_s},$$

then reduced back to C_s channels via a 1×1 projection before entering the `Detect` head.

Interface with the Detect head

The detection head expects three feature maps with original channel counts $\{C_s\}$. MaYOLOc achieves this by (i) concatenating $d = 64$ context channels at each scale, (ii) projecting $(C_s + d) \to C_s$, and (iii) passing the reduced features into the unmodified `Detect` head. Thus, the architecture remains fully compatible with the YOLO pipeline.

The figure 4.3 shows how the context module is injected between the backbone + neck and the head, with the attention module acting as a bridge.

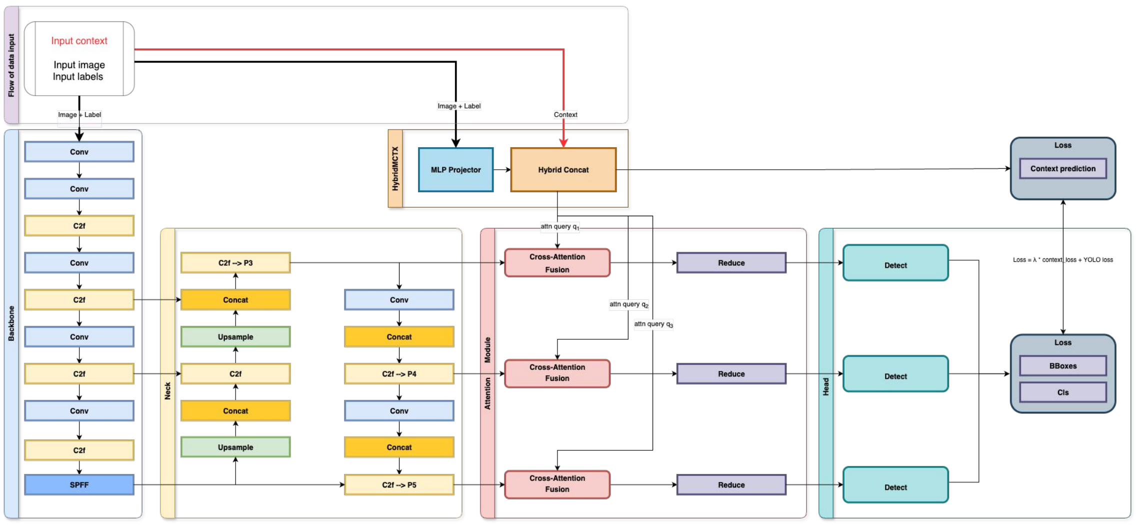


Fig. 4.3: Schematic overview of the MaYOLOc architecture.

4.2.4 Implementation Notes and Challenges

This section outlines the critical aspects and strengths of the proposed implementation, highlighting the main differences compared to the YOLO*C model [32] presented in the chapter dedicated to the state of the art and providing the reasons behind the design choices adopted in this work.

As explained in the model introduction, the implementation is designed to be compatible with various YOLO architectures from version 8 onwards, as long as they adopt the anchor-free paradigm. This has been possible thanks to the flexible hooking system capable of automatically determining the dimensionality of the last three high-level outputs of the neck and adapting the integration logic to the requirements of the selected model. This aspect represents the first improvement over the YOLO*C model, which was based on an older model (YOLOv5) that still relied on anchor-based detection. In MaYOLOc, the model selection is simple: just change the configuration variable corresponding to the desired YAML file, which specifies the architecture to use. This design choice provides a clean and manageable interface for switching between YOLO variants without changing the basic implementation. This flexibility is critical for two reasons: (i) since the Ultralytics repository is subject to frequent updates, even a small bug fix could alter the internal logic and potentially compromise an implementation, making a fallback mechanism to the previous version essential; (ii) the same mechanism ensures compatibility with future versions, allowing the framework to adapt to new and future versions of YOLO, thus keeping pace with the latest developments in the architecture.

However, the dynamic insertion and hooking strategy introduces a significant drawback. The integration of the context module and its subsequent connection to the detection head inevitably triggers a complete reinitialization of the weights associated with detection. As a result, the model effectively “loses” the knowledge previously acquired in classification and localization tasks, requiring retraining of the detection head almost from scratch. Emphasizing this pain-point is of fundamental importance because it represents one of the main limitations of the implementation, which, as we will see in Chapter 5, will also be reflected in the metrics.

Beyond architectural changes, both approaches extend the YOLO framework by incorporating an additional branch dedicated to context-based classification and by augmenting the training objective with an explicit context-related loss term. This design choice reflects a shared motivation: enhancing detection performance through the integration of contextual information while preserving the efficiency of a one-stage detector. YOLO*C employs the Multi ConTeXt (MCTX) module, which concatenates multi-scale feature maps at the smallest resolution, applies a dimensionality reduction via a 1×1 convolution, and subsequently flattens the output before classification through fully connected layers. This design remains purely convolutional and does not exploit attention mechanisms. By contrast, MaYOLOc integrates a HybridMCTX module, which combines convolutional operations with attention-based mechanisms, thereby enabling the model to capture more complex dependencies between context variables and visual features.

Injecting the context module at the level of the neck outputs (P3–P5) rather than within the backbone reflects a deliberate trade-off between effectiveness and stability. Backbone layers primarily encode low-level primitives such as edges, corners, and textures, which must remain invariant across conditions; perturbing them with contextual information risks undermining generalization. In contrast, the neck produces semantically enriched, multi-scale representations that are closer to the detector's decision-making process. Conditioning these intermediate feature maps allows the model to refine higher-level cues relevant to objectness and classification, without interfering with the fundamental low-level structures on which the detection process depends. Furthermore, aligning the context injection with P3–P5 ensures that contextual information is explicitly propagated into the feature pyramid, thereby improving robustness to environmental variations while preserving YOLO's efficiency. Alternative strategies, such as injecting context deeper into the backbone or directly at the detection head, were considered but discarded, particularly given the constraints imposed by the availability of pre-trained weights.

To summarize the main architectural and functional differences between the two approaches, table 4.1 provides a side-by-side comparison of YOLO*C and the proposed MaYOLOc model. An important aspect that deserves careful consideration is the robustness of the model when dealing with missing or unreliable metadata. The hybrid mixing mechanism is designed to work at both the sample level and the level

Aspect	**YOLO*C** [32]	**MaYOLOc**
Base model	Built on YOLOv3, YOLOv5, YOLOv7 (anchor-based)	Compatible with YOLOv8+ (anchor-free)
Context module	MCTX: concatenates multi-scale features (smallest resolution), 1×1 conv for reduction, flatten + dense layers	HybridMCTX: convolution + attention-based mechanisms
Context injection	Auxiliary branch added after neck, parallel to detection heads	Injected into neck outputs (P3–P5), integrated with feature pyramid
Handling metadata	Requires complete and reliable context labels	Hybrid mixing, robust to missing/incomplete metadata (fallback predictor)
Flexibility	Bound to YOLOv3-YOLOv5-YOLOv7 implementation	Flexible hooking system across YOLOv8+ variants via YAML config
Limitations	Purely convolutional, no attention; sensitive to noisy/ambiguous context labels	Reinitializes detection head after context integration, requires retraining

Table 4.1: Comparison between YOLO*C and MaYOLOc

of individual context variables. For example, if sea state is available but visibility is missing, the final context vector c is constructed by combining the one-hot encoding of the known sea state with the posterior distribution predicted for visibility. This design ensures a form of gradual degradation: instead of failing when metadata is incomplete, the model seamlessly integrates the available information with internally inferred estimates. Such robustness is critical to avoid unstable behavior in real-world conditions, where metadata may be partially available, noisy, or recorded inconsistently. For now, the context predictor should be considered as an accurate and reliable fallback option, although its expansion is left to future work.

4.3 Training Pipeline

The training procedure of MaYOLOc is encapsulated in a dedicated trainer, designed to handle in an integrated manner all the stages required for model optimization, evaluation, and logging. The entry point of the process is the configuration, which provides a flexible interface for defining the training parameters such as the number of epochs, the batch size, the initial learning rate, the learning rate decay factor, and the optimizer's momentum schedule. These parameters are directly extracted from the training configuration and propagated through the entire pipeline, ensuring consistency and reproducibility of the experiments. The definition of the device on which the training is executed (CPU or GPU) is also managed at this stage, allowing the implementation to be portable across different computational environments.

The trainer also handles the dataset management. The pipeline makes use of the customized dataset builder discussed in sec.4.2.1, which returns the training and validation sets along with their respective data loaders. As already mentioned, certain data augmentation operations, such as mosaic augmentation, are disabled in order to avoid inconsistencies in the context logic. Both the training and validation dataloaders are configured to match the batch size and worker parallelism specified in the configuration, thereby ensuring an efficient feeding of data during training.

The optimization strategy follows a multi-phase schedule. The trainer defines an optimizer based on AdamW, with three distinct parameter groups: biases, attention-related parameters, and all remaining weights. This separation allows a differentiated learning rate schedule, where bias terms start from a lower warm-up learning rate, attention layers receive a reduced step size to stabilize training, and the remaining parameters follow the main learning rate schedule. The learning rate evolves according to a two-stage scheduling mechanism. During the initial iterations, a linear warm-up gradually increases the step size while simultaneously adjusting the momentum from an initial value to the target momentum. This phase is essential to prevent instability during the first training steps. Once the warm-up is complete, a cosine annealing schedule governs the learning rate decay across the remaining iterations, ensuring a smooth reduction toward a minimum value proportional to the initial learning rate. Both phases are chained together, and the scheduler is updated at each training iteration rather than at the epoch level, which allows for finer control of the optimization dynamics.

The model is trained under the hybrid loss function that combines the detection loss of YOLO with an additional context-aware component, as explained in sec.4.2.2. The detection loss is computed by the standard YOLO criterion, including bounding box regression, classification, and distributional focal loss, whereas the context loss supervises the prediction of metadata variables such as sea state, visibility, and acquisition channel. The relative weighting of these two terms is configurable and determines the strength with which contextual predictions influence the optimization.

The core of the trainer is the iterative training loop. For each epoch, the trainer iterates over all batches, preparing both the image data and their annotations, as well as the corresponding context variables. Missing metadata are handled explicitly by marking them with sentinel values, which are interpreted during the forward pass by the hybrid context mixer. The forward step produces a set of predictions including the detections, the logits for each context variable, and the internal context embedding used by the cross-attention modules. The loss is then computed and backpropagated, and the gradients are regularized by a clipping mechanism to prevent numerical instabilities. The optimizer step is executed at each iteration, immediately followed by the scheduler update. Logging during training is performed through a progress bar that reports in real time the GPU memory consumption, the values of the different loss components, and the learning rates of the parameter groups.

At the end of each epoch, the trainer switches to evaluation mode and computes validation metrics. The evaluation pipeline leverages the TorchMetrics library to calculate the mean Average Precision (mAP) over IoU thresholds from 0.5 to 0.95, together with recall and maximum average recall at multiple cutoffs. Predictions undergo non-maximum suppression to ensure consistency with YOLO evaluation standards, and both predictions and ground truths are converted into the format expected by the metrics library. Importantly, the accuracy of the context predictions is also evaluated, considering only those samples where the ground truth metadata are available. This dual evaluation provides insight not only into the detection capabilities of the model but also into the effectiveness of the context predictor.

All results are systematically recorded. After each epoch, the trainer logs into a CSV file the evolution of the losses, the main detection metrics, and the learning rates of the different parameter groups. Two plots are then generated: one showing the dynamics of the losses and metrics across epochs, and another representing the evolution of the learning rate schedule. Model checkpointing follows a rigorous strategy: the best-performing model, defined as the one that maximizes a fitness score based primarily on mAP@0.5–0.95 with a smaller contribution from mAP@0.5, is saved separately, while the last model is always stored to ensure recoverability of the training process.

4.3.1 Training Environment

Some of the experiments performed with the baseline Ultralytics YOLO model were executed on the Leonardo supercomputer, specifically on the Davinci-1 cluster, one of the most advanced high-performance computing (HPC) infrastructures currently available. Exploiting such a resource required a thorough study and adaptation of the training pipeline to the HPC environment, in particular through the use of Singularity, a containerization framework specifically designed for HPC systems. Unlike Docker, which is not natively supported on most supercomputing clusters due to security restrictions, Singularity provides operating-system-level virtualization in user space, ensuring both reproducibility of experiments and compatibility with the HPC scheduler. The entire training environment was encapsulated into a Singularity Image File (SIF). This container bundled all required components: Python execution scripts, the Ultralytics YOLO framework, dependencies such as PyTorch with GPU acceleration (CUDA and cuDNN), custom dataset directories, and preconfigured environment variables. By encapsulating the environment, portability was guaranteed, eliminating inconsistencies that might arise from software version mismatches between local development machines and the HPC execution nodes. The SIF was generated from a Singularity definition file, which specifies in a declarative manner how the container environment should be constructed.

Job submission to the cluster was handled through the Portable Batch System (PBS) scheduler, which manages resource allocation across thousands of simultaneous users. A dedicated PBS script was written to define all the job requirements and submission parameters. This script specified resource allocation, as the number of nodes, number of CPUs per node, number of GPUs, total walltime (execution time limit), and memory allocation, but also the redirection of output and error logs or other project specification. The workflow was orchestrated through SSH-based job submission, where the researcher prepared the PBS script locally and submitted it via the command line interface. Once the job was queued and subsequently scheduled on a suitable node, the container image was loaded on the node's local storage, and training was executed in full isolation inside the Singularity environment. In terms of hardware, davinci-1 offers several possible computing nodes. I had the opportunity to use a GPU computing node with a 2.80 GHz CPU (24 cores each), 512 GB of RAM, and up to 4 NVIDIA A100 GPUs with 40 GB each.

While the opportunity to exploit such high-performance resources is undoubtedly valuable, the process of building the container image and porting the entire environment to the HPC infrastructure proved to be in tension with the flexibility and iterativeness required during the development of the MaYOLOc model. Containerization and deployment to the cluster introduce an additional layer of overhead, which is well suited for stable and large-scale experiments but less compatible with the rapid prototyping cycle typical of research and model design. For this reason, during the development phase the experiments were primarily conducted in a local environment, where changes could be tested and validated more quickly, while the export of the workflow to HPC was left for future stages, once the model architecture reaches greater maturity.

Therefore, locally it has been possible to use an high-performance workstation designed to support computationally demanding tasks. The system is built on an ASUSTeK COMPUTER INC. WS X299 SAGE platform, equipped with an Intel® Core™ i9-9920X processor providing 24 logical cores, and 128 GiB of system memory. For parallelized deep learning operations, the workstation is furnished with four NVIDIA GeForce RTX™ 2080 Ti GPUs, enabling efficient large-scale training and experimentation. Storage requirements were supported by a 1 TB disk capacity, ensuring sufficient space for datasets, intermediate computations, and model checkpoints. This hardware configuration provided the computational resources necessary to handle both the training and evaluation of the implemented models in a reliable and efficient manner.

Chapter 5

Model Evaluation

Object detection performance in YOLO is assessed with a set of quantitative metrics that capture both the accuracy of identifying objects and the precision of their localization. These metrics provide insight into how effectively the model detects objects in images and how well it balances errors like false positives (detecting objects that are not there) and false negatives (missing real objects).

This chapter is structured as follows. First, we present the methods used to calculate the evaluation metrics typically employed in YOLO, along with an additional metric designed to evaluate the accuracy of contextual variable predictions. Next, we shift our focus to the experiments conducted on custom datasets (chap.2) and the corresponding results. Finally, there will be a section on the explainability of the model in order to discuss and contextualize the results.

5.1 Metrics

At the core of YOLO's metric computation is Intersection over Union (IoU), a fundamental measure of overlap between the predicted bounding box and the ground truth box for an object. The IoU is defined as the area of overlap divided by the area of union of the predicted and true bounding boxes, yielding a score between 0 and 1 (with 1 indicating perfect alignment). In terms of formulas, the overlap between a predicted bounding box B_p and a ground-truth bounding box B_g is quantified by

$$\mathrm{IoU}\left(B_p, B_g\right) = \frac{|B_p \cap B_g|}{|B_p \cup B_g|}, \tag{5.1}$$

where $| \cdot |$ denotes the area of the region. YOLO uses IoU to determine whether a predicted detection counts as a True Positive. In practice, a detection is considered correct if its IoU with a ground truth object exceeds a given threshold (for example, 0.5, which is a common default). Detections that fall below the IoU threshold or that correspond to no ground truth are treated as False Positives. Unmatched ground-truth objects that have no corresponding prediction are counted as False Negatives. By applying this IoU criterion, YOLO ensures that predicted bounding boxes have sufficient overlap with the actual objects before they are credited as correct detections, thereby incorporating localization accuracy into the evaluation.

Building on the IoU-based matching of predictions to ground truth, YOLO computes the classical Precision and Recall metrics to summarize detection performance. Precision is defined as the fraction of predicted objects that were actually correct detections (i.e. true positives divided by all positive predictions). A high precision means that the model has a low false alarm rate, indicating it rarely misidentifies background or incorrect regions as objects. Recall, on the other hand, is the fraction of all actual objects that were successfully detected by the model (true positives divided by the total ground-truth objects). An high recall means few objects were missed by the detector. Let TP denote the number of true positives, FP the false positives, and FN the false negatives. Then:

$$\begin{aligned} \text{Precision } &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \\ \text{Recall } &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \end{aligned} \tag{5.2}$$

There is typically a trade-off between precision and recall: a model can achieve higher recall by predicting more objects (including some false alarms), but this may reduce precision. To evaluate the balance between these two factors, the F1 score is often considered, which is the harmonic mean between precision and recall. An high F1 score indicates that the model achieves a good balance, detecting a high proportion of objects while keeping false positives low. The inclusion of precision, recall, and F1 metrics helps clarify the error profile of the model, for example, whether it is overly conservative (high precision, low recall) or overly aggressive (low precision, high recall) in its detections.

$$F_1 = \frac{2 \cdot \text{ Precision } \cdot \text{ Recall}}{\text{Precision } + \text{ Recall}}. \tag{5.3}$$

While precision and recall at a fixed operating point are useful, a more comprehensive metric is needed to summarize performance across all possible confidence thresholds. YOLO relies on the Precision-Recall (PR) curve and the derived Average Precision (AP) to perform this role. For each object class, the model's predictions can be sorted by their confidence score from high to low. By gradually lowering the confidence threshold, one can compute the precision and recall at each step: initially, only the top-most, very confident detections are considered (yielding high precision but possibly lower recall), and eventually, even low-confidence predictions are included (increasing recall but potentially lowering precision). Plotting precision against recall as the threshold varies produces the PR curve for that class. For each class c, at each rank k, precision and recall are computed as in eq.5.2, giving a curve:

$$\mathcal{P}_c(r) \quad \text{vs.} \quad r \in [0,1], \tag{5.4}$$

Average Precision (AP) is then computed as the area under this precision-recall curve. It effectively integrates the model's performance across all recall levels, providing a single number that captures the trade-off between precision and recall for that class. The AP for class c at IoU threshold τ defined as the integral of the precision–recall curve:

$$AP_c(\tau) = \int_0^1 \mathcal{P}_c(r;\tau)dr \tag{5.5}$$

In discrete implementations, this is computed as a numerical sum over recall points:

$$AP_c(\tau) = \sum_{n=1}^{N} (r_n - r_{n-1})\, \mathcal{P}_c\,(r_n;\tau) \tag{5.6}$$

where r_n is the recall at the n-th threshold. Conceptually, AP can be thought of as the average of precision values over the range from 0% to 100% recall. The result is a robust metric for each class: an AP of 1.0 (or 100%) would indicate perfect detection for that class (all objects found with no false positives), whereas lower AP values signify the extent to which detections are missed or spurious. Because AP encapsulates performance at all confidence levels, it is widely regarded as a gold-standard metric for object detection models.

For datasets with multiple object classes (which is the typical scenario), YOLOv8 extends the AP concept to Mean Average Precision (mAP). The mAP is simply the mean of the AP values across all classes in the dataset. This averaging treats each class equally, regardless of the number of instances, thus providing a balanced overview of performance across diverse object categories. A high mAP indicates that the model performs well on average for all classes, whereas a low mAP can reveal that the model is generally struggling or that performance is inconsistent (some classes may be detected well but others poorly). YOLO's validation tool reports the mAP alongside the per-class AP, allowing researchers to inspect both the aggregate performance and the class-specific details.

A key aspect of YOLO's evaluation methodology is the use of multiple IoU thresholds when calculating average precision, in line with the latest COCO evaluation standards. In addition to the classic AP computed at a single IoU threshold (usually 0.5), YOLO calculates what is often referred to as mAP@0.5:0.95. This is the mean of AP values computed at ten different IoU thresholds ranging from 0.50 to 0.95 in steps of 0.05. The mAP@0.50 can be seen as a metric of general detection ability, whole the mAP@0.50:0.95 (often just called the COCO-style mAP) is a much stricter and more thorough metric. Because it includes high IoU requirements (up to 95% overlap), it demands very precise localization for a detector to score well. This makes mAP@0.50:0.95 a comprehensive evaluation of both detection and localization quality. A model that scores high on mAP@0.5 but low on mAP@0.75

or mAP@0.95, for instance, may be finding the correct objects but not placing tight bounding boxes on them; by averaging over many IoU levels, the mAP@0.50:0.95 metric will reflect this shortcoming. YOLO therefore reports both metrics.

The validation output typically includes a table of per-class metrics (Precision, Recall, AP50, AP50-95 for each class) and the aggregate mean values across classes. In addition to numeric metrics, YOLO often provides visualizations such as the precision-recall curve and F1-vs-threshold curve for the whole model and even a confusion matrix illustrating the distribution of correct vs incorrect predictions across classes. These are complementary to the core metrics: for example, the PR curve can show if precision drops off sharply at a certain recall, and the confusion matrix can reveal if certain classes are being confused with others.

Since the entire training and validation cycle has been rewritten, the evaluation of the MaYOLOc model has also been built on the YOLO framework with the aim of reproducing the calculation of these metrics as closely as possible. The code leverages the torchmetrics module with configuration choices that align with the Ultralytics YOLO standard. The module internally computes precision–recall curves for each class, and the average precision (AP) is then integrated using the 101-point interpolation method. To remain consistent with the COCO evaluation protocol, AP is calculated at multiple Intersection-over-Union thresholds ranging from 0.50 to 0.95 in steps of 0.05, yielding the standard mAP@0.50:0.95 score. In addition, the implementation reports mAP@0.50, allowing for direct comparison with legacy YOLO metrics. Beyond conventional detection metrics, the MaYOLOc framework introduces an additional evaluation dimension for context prediction, totally unrelated from the detection metrics. To quantify the accuracy of the predictions of the context variables, the code maintains a `context accuracy` dictionary, computing classification accuracy for each context variable independently. The accuracy is defined as:

$$\text{ContextAccuracy}\ (v) = \frac{\text{Correct Predictions of } v}{\text{Total Predictions of } v}, \tag{5.7}$$

where v denotes a particular contextual attribute. While it does not directly affect the bounding box–based mAP computation, it complements the detection evaluation by quantifying performance in the auxiliary prediction task introduced by the MaYOLOc architecture.

5.2 Experiments and Results

Since this thesis has a specific purpose beyond pure research and development, experiments parallel to those of the presented model were carried out during the project to ensure that Leonardo can still count on a semi-ready output. For this reason, it will be first presented the experiments and the results of the YOLO models on the two custom datasets, and then the performance of the MaYOLOc model.

5.2.1 YOLO

After initial training experiments on the Coast Guard dataset, we felt that additional data was needed, so an external source was introduced to expand the instances while remaining consistent with the maritime domain. For this reason, the MOBDrone dataset, introduced in chapter 2, was incorporated.

The dataset had a dual role: on the one hand, it served as a benchmark for assessing the extent to which the model's performance depended on the quality and characteristics of the Coast Guard data; on the other hand, it represented the first stage of a two-stage tuning process designed to precede the final adaptation to the Coast Guard dataset. This last experimental design was motivated by several factors. First, the CG dataset on its own is too limited in scale to support robust training, making it necessary to leverage additional data sources. Second, in section 3.1.4 we proved that transferring from general-purpose pretrained models, such as those trained on COCO, to a highly vertical task often yields suboptimal results, reinforcing the need for a domain-aligned dataset such as MOBDrone.

However, the inference results obtained on the Coast Guard dataset from the model previously fine-tuned on MOBDrone were not significant in terms of performance. Unexpectedly, increasing the amount of data with such a large and different dataset from that of the Coast Guard proved to be disadvantageous. Table 5.1 shows a comparison of the inference metric results obtained on MOBDrone, on Coast Guard dataset, and Coast Guard previously fine-tuned on the MOBDrone.

Dataset	**Precision**	**Recall**	**mAP@50**	**mAP@.50:.95**
MOBDrone	0.91	0.809	0.86	0.509
Coast Guard	0.812	0.675	0.715	0.397
MOBDrone + CG	0.726	0.66	0.684	0.374

Table 5.1: Comparison of the YOLOv11s inference results in different experimental combination

Therefore, the YOLOv11 model pre-trained on COCO and fine-tuned directly on Coast Guard seems to outperform the same model that has undergone an additional fine-tuning step on MOBDrone. This is presumably due to two reasons: (1) given the large amount of data in MOBDrone, the model tends to "overfit" on those instances, losing the generality obtained from COCO training and, consequently, (2) the difference in terms of quality, image clarity, and diversity of acquisition scenes between the data in MOBDrone and that of the Coast Guard does not allow the model to learn correctly in the last fine-tuning phase.

Trying to solve this problem, we worked further on the MOBDrone dataset, applying augmentation processes that paradoxically aimed to "dirty" and "worsen" the frames. In addition, we decreased the number of samples and created a dataset that combined high-quality images with images that had undergone pixel-level modifications (such as "salt and pepper" noise, blur, motion). This process of making the quality of the two frames similar aimed to improve the ability of generalization of the model and to avoid its overfit on clean and unrealistic frames. Being MOB-

DroneV2 the second modified version of the MOBDrone dataset, the table 5.2 shows the comparison results of the same model with different combination of dataset and fine-tuning.

Dataset	Precision	Recall	mAP@50	mAP@.50:.95
MOBDroneV2	0.693	0.608	0.641	0.323
MOBDrone	0.91	0.809	0.86	0.509
MOBDroneV2 + CG	0.673	0.624	0.623	0.333
MOBDrone + CG	0.726	0.66	0.684	0.374
Coast Guard	0.812	0.675	0.715	0.397

Table 5.2: Comparison of YOLOv11n inference results in different experimental combination after MOBDrone dataset modification

As shown, the results are not yet very promising, leading us, contrary to expectations, to abandon the idea of using the MOBDrone dataset as the first step in a dual fine-tuning process. For this reason, attention has shifted to maximizing the results obtainable from fine-tuning YOLO directly on the Coast Guard dataset.

In doing that, we experimented some inference technique as like SAHI (Slicing Aided Hyper Inference) [1]. SAHI is a method designed to improve object detection, particularly for small or distant targets, by dividing each image into overlapping slices and running inference on them separately before aggregating the predictions. In addition, we did not limit ourselves to applying the SAHI method only during inference. The image cropping mechanism was also integrated into the training process, thereby increasing the number of data instances by generating multiple crops for each image. This enhancement strategy was expected to improve the model's ability to detect crops by mitigating the false positives that typically arise when SAHI is applied solely during inference. The anticipated outcome was a more balanced training process and, consequently, an overall improvement in evaluation metrics. However, as table 5.3 highlight, the experimental results did not confirm this expectation, likely due to the quality of the available data and the specific characteristics of the bounding boxes, which also contains medium-to-large bounding boxes. In line with the overarching objective of developing a flexible and effective tool applicable to a variety of scenarios, the model obtained through standard YOLO training was ultimately preferred.

Dataset	Precision	Recall	mAP@50	mAP@.50:.95
Coast Guard	0.812	0.675	0.715	0.397
Coast Guard w/ sahi	0.659	0.51	0.549	0.31

Table 5.3: Comparison of YOLOv11n inference results between the Coast Guard dataset with and without SAHI-like cropping in the training dataset

The evaluation results reported in table 5.3 indicate an overall solid performance of the model, with a precision of 0.812 and a recall of 0.675, leading to a mAP@50 of 0.715. This suggests that the detector achieves a good balance between precision and recall, though it tends to be conservative in its predictions, favoring precision over coverage. The drop to 0.397 in mAP@50:95 highlights limitations in bounding-box localization, with predictions often sufficient at lower IoU thresholds but less accurate when stricter alignment is required.

A class-wise analysis (Appendix, Figures A–F 1 and table 5.4) reveals heterogeneous behavior across categories. The Boat class achieves the highest reliability ($AP = 0.923$), with consistently high precision and recall across confidence thresholds. Person overboard reaches a satisfactory AP of 0.802 but shows reduced stability at higher confidence values, reflecting occasional confusion with background regions. In contrast, the Person class suffers from weak separability (AP = 0.420), as confirmed by the confusion matrices, where a large proportion of persons are misclassified as background or boats. This discrepancy is consistent with the limited sample size and the visual similarity of small-scale human figures with sea clutter, both of which hinder generalization.

The confidence-dependent F1, precision, and recall curves further suggest that the optimal operating point is reached at relatively low confidence thresholds (≈ 0.25), beyond which recall declines rapidly while precision continues to rise. This behavior implies that for operational use, threshold selection must be carefully tuned according to the application requirements: lower thresholds if recall is critical (e.g., safety monitoring), or higher thresholds if minimizing false alarms is prioritized.

Class	**Precision**	**Recall**	**mAP@50**	**mAP@.50:.95**
All	0.821	0.675	0.715	0.397
Boat	0.899	0.891	0.923	0.703
Person	0.613	0.361	0.42	0.21
Person overboard	0.923	0.773	0.802	0.278

Table 5.4: Per class metrics analysis of the best model of Coast Guard dataset - YOLOv11s

In terms of training parameters and hyperparameters, the best model was obtained with the configuration evidenced in table 5.5. Note that the number of training epochs is adjusted by the patience parameter, which implements early stopping by terminating the training process if no improvement is observed in a predefined number of consecutive epochs. In addition, the final learning rate reported does not represent the absolute value of the learning rate at the last epoch, but it corresponds to a multiplicative decay factor that progressively reduces the initial learning rate over the course of training.

Parameters	**Value**	**Parameters**	**Value**
Epochs	200	**Weight Decay**	0.005
Batch size	16	**Momentum**	0.937
Image size	960	**Warmup epochs**	10
Learning Rate	0.0001	**Optimizer**	AdamW
Learning Rate (final)	0.1		

Table 5.5: Training parameters and hyperparameters of the best model - YOLOv11s

Training experiments were also conducted using earlier versions of the architecture (YOLOv8). The results were consistent with the performance trends reported by Ultralytics, thereby confirming YOLOv11 as the most effective model currently available for object detection. Regarding model size, the small ("s") variant was identified as the optimal compromise between predictive performance and computational efficiency. Moreover, all training runs presented in this section were carried out under comparable conditions, with consistent training parameters and hyperparameters to ensure a fair evaluation.

5.2.2 MaYOLOc

The training process for the MaYOLOc model was comparatively less diversified, as context data were generated only for the Coast Guard dataset. Before proceeding with the analysis of the results, it is essential to recall the implementation notes already outlined in Section 4.2.4, which inevitably limit the model's ability to compete with state-of-the-art approaches at this stage.

The purpose of this experiment, however, is not to establish MaYOLOc as a competitive alternative to existing benchmarks, but rather to investigate the feasibility of integrating contextual information through an attention mechanism. The main objective is to assess whether the model demonstrates effective learning and, in particular, how the added attention module contributes to this process. The following discussion will therefore not be limited to interpreting the results obtained, but will also contextualize them and place them within this exploratory framework. In the next section, the analysis will be extended to the domain of explainability, with the aim of clarifying the role of contextual information in shaping the learning models underlying the model's behavior.

Dataset	**Precision**	**Recall**	**mAP@50**	**mAP@.50:.95**
Coast Guard w/ context	0.36	0.68	0.3	0.18

Table 5.6: MaYOLOc inference results

The inference results obtained with MaYOLOc (Table 5.6) highlight both the potential and the limitations of this experimental architecture. The recall value (0.68) is essentially equivalent to that of the baseline YOLO model (0.675), which is remarkable given the constraints imposed during training. Precision, on the other hand, drops sharply to 0.36 compared to 0.812 in the baseline, confirming that the model struggles to effectively suppress false positives under the current setup. A similar trend is observed in the mAP values: at IoU 0.5, performance decreases from 0.715 to 0.30, while at IoU 0.50:0.95 it falls from 0.397 to 0.18 ($\approx 50\%$ reduction).

The substantial reduction in accuracy and mAP is largely attributable to the reinitialization of the detection head weights, which limited the model's ability to converge to accurate bounding-box regression and classification. However, this limitation does not obscure the fact that the model is already exploiting contextual information to support recall, which proportionally to the value of precision is particularly high. With a richer training set and more extensive optimization, the precision and mAP values could recover, while recall could improve further, thus improving the overall effectiveness of the approach.

However, this reduction should not be regarded as the central outcome of the experiment. Rather, it reflects the provisional nature of the implementation, which is limited at this stage by the cons inevitably introduced that substantially affected the detection accuracy, especially given that attention mechanisms typically require larger datasets and more extensive training to reach their full potential. A visual inspection of the model during inference also reveals that although the detection performance is generally satisfactory, the behavior still resembles that of a model at an early stage of training. The predictions appear consistent, but lack the robustness and stability typically observed in fully convergent models, suggesting that further training is needed to refine its representational maturity.

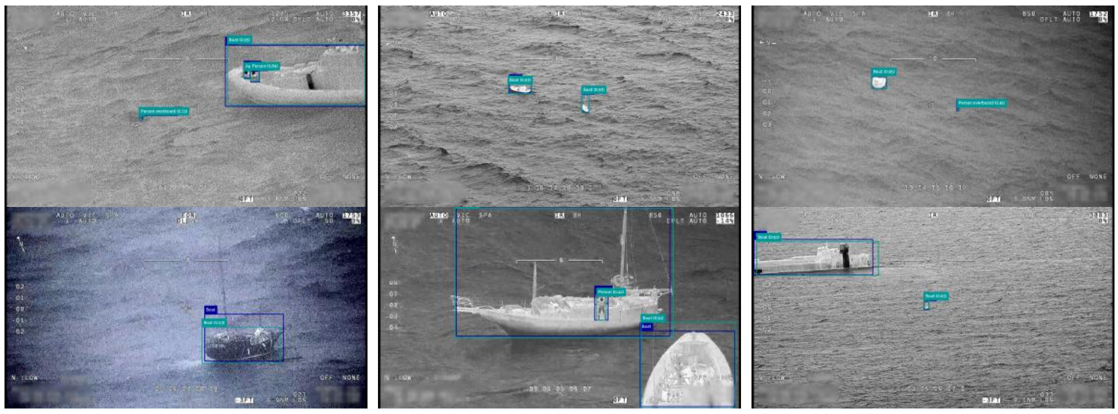


Fig. 5.1: Mosaic of MaYOLOc inference sample detection - in blu ground truth, in azure the prediction of the model

Within this context, MaYOLOc should be understood as a proof of concept, designed to test the integration of contextual information into the detection pipeline and to provide insights into its potential benefits. Its value lies less in current quantitative performance than in demonstrating the viability of this approach and establishing a foundation for future iterations that will incorporate more extensive data, augmentation strategies, and training refinements.

In terms of training parameters, it is important to note that, unlike the YOLO model discussed in the previous section, the number of warm-up epochs was increased to facilitate a proper initialization of the attention module. This adjustment ensures that the weights are stabilized during the initial training phase, when a lower learning rate is applied, before reaching a maximum value of 0.001. The learning rate itself is not fixed but dynamically adapts across different layers. Specifically, a differentiated learning rate strategy was employed, assigning distinct values to the convolutional layers, biases, and attention layers. As in the YOLO framework, the bias parameters are optimized using the ”warmup bias lr” parameter, which is set by default to 0.1. In contrast, the attention modules are trained with a learning rate equal to 0.4 times the base value specified in the configuration.

Parameters	**Value**	**Parameters**	**Value**
Epochs	300	**Batch size**	16
Image size	960	**Context weight**	0.05
Learning Rate	0.001	**Warmup bias lr**	0.01
Learning Rate (final)	0.1	**Warmup epochs**	16
Weight Decay	0.005	**Optimizer**	AdamW
Momentum	0.937		

Table 5.7: Training parameters and hyperparameters of MaYOLOc

5.3 MaYOLOc explainability

Moving beyond aggregate performance measures, it is necessary to examine how the added attention block operates internally and whether it contributes meaningfully to the learning process. This is addressed through an explainability analysis that quantifies the behavior and contribution of the attention mechanism across three dimensions:

- Entropy of the attention distribution: assessing the sharpness or diffuseness of attention allocation across spatial positions.
- L2 norm of the context embedding: measuring the effective magnitude of the contextual signal injected into the feature maps.
- Gradient flow through attention-related layers: evaluating the propagation of

learning signals to verify that the attention block is trainable and actively updated.

These diagnostics do not aim to explain model predictions in a human-interpretable sense, but rather to provide quantitative evidence of whether the attention mechanism is functioning as intended. Entropy characterizes the statistical structure of the attention weights, the embedding norm indicates the strength of contextual modulation, and gradient flow captures the capacity of the module to receive and transmit updates during training.

5.3.1 Entropy of the Attention Distribution

Entropy is measure of the uncertainty inherent in a probability distribution. For a discrete distribution $A = (a_1, a_2, ..., a_N)$ with $\sum_{j=1}^{N} a_j = 1$ and $a_j \geq 0$, the entropy is defined as:

$$H(A) = -\sum_{j=1}^{N} a_j \log a_j \tag{5.8}$$

This scalar quantifies the uncertainty or spread of the distribution. Applied to attention mechanisms, A corresponds to the normalized attention weights across the $N = H\text{x}W$ spatial positions of a feature map. In this context, entropy helps us understand how uniformly distributed the weights of attention are in the spatial domain. Heuristically, transforming entropy values into a spatial heatmap would mean understanding "on which point of the image the model is focusing its attention". Having three different features maps inputting the attention module (P3, P4, P5), the attention entropy will be computed per feature level per image.

In order to interpret the values correctly, we distinguish between two extreme cases. For a 960x960 input image, the P3 features maps is 120x120. Therefore, the upper bound is defined as the $log(H \times W) = log(120 \times 120) \simeq 9.57$. When the entropy reaches these values, it is a near-uniform attention (no focus) and it does not really contribute. On the other hand, when the entropy stays very low (from 0 to 2-3) the attention is collapsing, and again it does not contribute to the model detection.

In addition to the scalar entropy values, the spatial distribution of the attention weights was inspected directly through heatmaps projected onto the feature maps. These visualizations provide a qualitative counterpart to the quantitative analysis, making it possible to observe where the context-conditioned query allocates focus within the feature hierarchy. While entropy summarizes the distribution in a single value, the heatmaps reveal its structure—whether attention is concentrated in compact regions, spread across multiple salient areas, or dispersed uniformly. The following plots therefore serve to contextualize the numerical results by illustrating typical patterns of attention allocation at different feature scales. Figure 5.2 shows different heatmaps at different scales (P3, P4, P5) and therefore different dimensions.

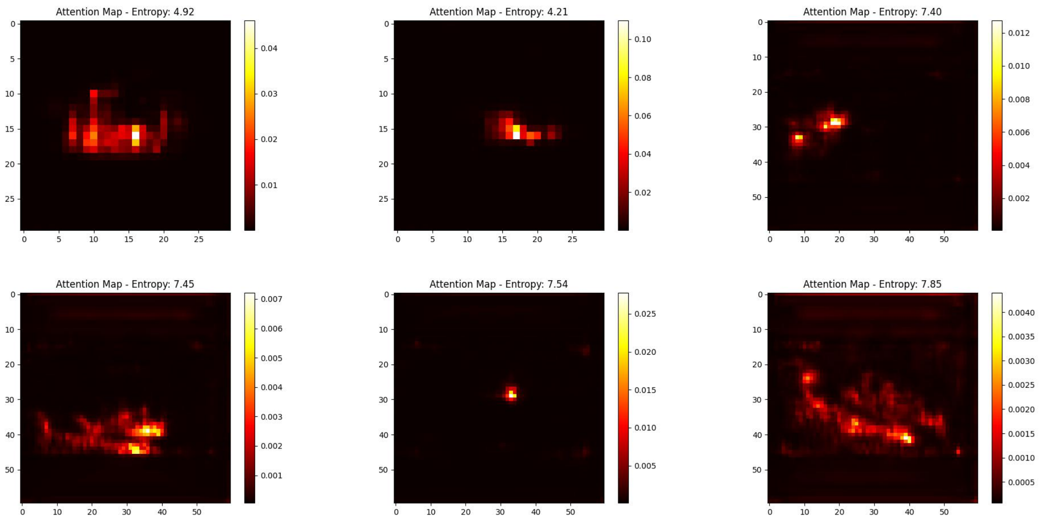


Fig. 5.2: Mosaic of attention entropy heatmaps of sample frames

The analysis of the entropy distribution becomes even more significant when compared with the original image that the model takes as input (fig. 5.3)
In this representative example, it can be observed that the model successfully directs its attention toward the object of interest with remarkable precision. The entropy map highlights a clear concentration of information around the target, demonstrating the model's ability to suppress irrelevant background features. The feature map

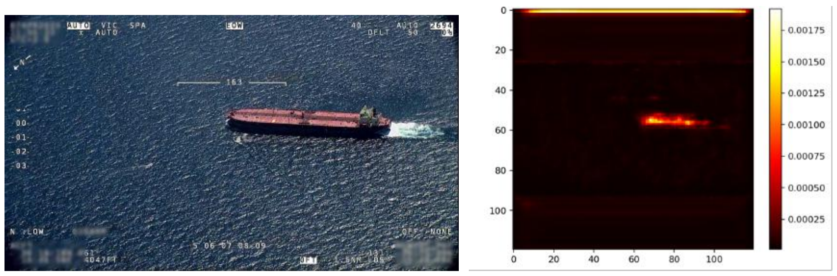


Fig. 5.3: Comparison between original image and entropy output on feature map (P3)

under examination corresponds to the P3 level, which is the highest-resolution feature map within the detection head(120×120). As such, it preserves a greater degree of spatial detail and structural similarity with the original image, making the correspondence between attended regions and visual content especially evident.

5.3.2 Quantifying Embedding Contributions and Gradient Flow via the L2 Norm

A critical aspect of understanding the trainability of attention-based architectures lies in quantifying the magnitude of the signals that propagate through the embedding space and across parameter gradients. Since embeddings and gradients are inherently high-dimensional objects, interpreting individual coordinates offers little insight. Instead, it is necessary to employ scalar proxies that summarize their overall contribution strength. The L2 norm, also known as the Euclidean norm, provides a mathematically rigorous measure for this purpose. Formally, for a vector $v \in R^d$ with components $(v_1, v_2, ..., v_d)$ the L2 norm is defined as:

$$\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^{d} v_i^2} \tag{5.9}$$

The L2 norm treats the embedding vector as a point in Euclidean space and measures its overall strength in a coordinate-independent way. Such rotational invariance is especially relevant for neural embeddings, since they are not tied to any "physical" coordinate system, so changing basis shouldn't change the notion of contribution. Moreover, the L2 norm is differentiable everywhere, which ensures smooth diagnostic behaviour even when individual vector components approach zero.

Context Embedding Diagnostic

When applied to the context embedding vector projected into the attention block, the L2 norm reflects the effective strength of the contextual signal injected into the fused representation. If the norm assumes very small values, the context branch contributes negligibly, which may indicate gradient starvation or a "dead" attention pathway. Conversely, excessively large values suggest unstable or exploding activations, which can dominate feature fusion and destabilize optimization. Ideally, the embedding norm should remain within an intermediate range, signifying that contextual information is consistently incorporated without overwhelming the visual features.

Empirical observations confirm that the L2 norm of the context embedding vector gradually increases over the course of training, providing evidence that the contextual branch becomes progressively more active and engaged in the learning process. Rather than collapsing toward zero, which would indicate an inactive pathway, the values remain consistently non-zero, demonstrating that contextual information is being transmitted into the fused representation.

Gradient Flow Diagnostic

The same principle applies when examining gradient vectors associated with the parameters of the attention block. The L2 norm of these gradients quantifies the total strength of the learning signal directed toward a parameter group. Large values indicate strong corrective pressure, while near-zero values imply insufficient updates, characteristic of vanishing gradients. Ideally, gradients should exhibit stable, non-zero magnitudes across all subcomponents of the attention mechanism, ensuring that the block is an active participant in the optimization process.

The empirical inspection of gradient flow confirms that the attention module is indeed learning in a structured and stable manner. Among the subcomponents, the query–key–value convolution weights exhibit the largest gradient magnitudes, often ranging from several tens up to the low hundreds before gradient clipping is applied. This dominance is expected, as these parameters govern the transformation of feature maps into the latent query, key, and value spaces, thereby controlling the core representational power of the block. The output projection convolution receives gradients of slightly smaller but still substantial magnitude, indicating that the module is actively refining how attended features are reintegrated into the representation.

It is important to note that although the raw gradient norms per parameter occasionally peak at the beginning of training, the combination of gradient cutoff and reduced learning rates for attention parameters ensures numerical stability and prevents divergence. Over epochs, a trend emerges in which gradient magnitudes gradually decrease from the initial peaks, while remaining non-zero. Overall, gradient propagation demonstrates that attention blocking is actively trainable and well-adjusted: it contributes significantly to the optimization process without exhibiting vanishing or exploding behaviors.

Chapter 6

Conclusions

This final chapter brings together the main outcomes of the research, following the progression outlined in the earlier chapters. Building on the review of state-of-the-art models presented in Chapter 2, the analysis of the dataset in Chapter 3, and the architectural design described in Chapter 4, the evaluation results of Chapter 5 are now reconsidered in light of the overarching objectives of the thesis. In addition to summarizing the research findings, the chapter outlines possible directions for future investigations, suggesting how the proposed model could be extended and refined.

Following an in-depth analysis of the state of the art, single-stage detectors emerged as the most effective solutions currently available, with YOLO and its variants standing out in particular. At the same time, hybrid models that integrate additional mechanisms showed promising potential, although they still require further research.

Based on these insights, the MaYOLOc model was developed as an extension of YOLO, incorporating an attention module positioned between the backbone-neck feature extraction and head detection stages. The purpose of this module is to improve the detection process by combining the features extracted by YOLO with additional contextual information related to the overall frame environment. Unlike most hybrid approaches, which typically introduce transformers directly into the detection phase, MaYOLOc retains YOLO's native detection pipeline. This design ensures the benefits of contextual integration without compromising the speed advantage that characterizes single-stage detectors.

The experiments were conducted mainly on the Coast Guard dataset, the production of which is part of the work carried out in this thesis. This dataset represents a valuable resource that will be further exploited internally at Leonardo for future research in the maritime domain. Its creation involved a structured process, beginning with video framing, followed by the filtering and preprocessing of frames to generate a collection of images. These images were subsequently annotated in YOLO format, partly through automatic inference and partly via manual labeling using the Label-Studio platform. In addition, to enable the use of the MaYOLOc model, contextual data was generated through prompt engineering and output redirection applied to a large language model (LLM). This process produced variable–value pairs representing the contextual information associated with each frame. The Coast Guard dataset and the additional MOBDrone dataset were part of the experiments conducted during the experimental phase. Although not central to the research objectives, the use of the MOB dataset proved highly informative. In particular, it highlighted the critical importance of working with data that are internally consistent. The detailed analysis of its composition and distribution, conducted in the initial phases, provided key insights that allowed us to interpret the models' performance more accurately - insights that would otherwise have remained obscured.

At this stage, we collected the results of the models. Given the profound innovation of the proposed approach, the developed model represents not only a concrete contribution to the field but also a valuable research stimulus for exploring the integration of contextual variables through the attention module. For this reason, it was equally important to conduct experiments with state-of-the-art models, in order to establish a solid benchmark for comparison. The outcomes of these experiments provide Leonardo with results that are immediately actionable, ready to be tested and implemented in operational scenarios, while MaYOLOc opens promising directions for future advancements in maritime object detection.

In light of the results obtained, the optimal strategy identified is the development of an on-demand solution, structured in two successive phases. In the initial stage, the YOLOv11 model, fine-tuned to maximize performance, is adopted to provide an immediately functional tool. This ensures that Leonardo can already deploy a reliable solution in operational contexts. At the same time, the continued collection of additional data will support the progressive refinement of the MaYOLOc model, bridging the gap arising from the limitations discussed in section 4.2.4 and thus improving model performance. The intention to persevere in developing this solution is fueled by the findings obtained in the explainability phase of the model and by the promising results achieved despite the significant limitations that the model entails.

6.1 Future Developments

The future directions of this project have already been partially discussed in the previous section. The primary objectives concern the collection of additional data to expand the dataset, together with the optimization and validation of the proposed model. To advance in this direction, it is essential to design a structured data collection system capable of acquiring contextual information directly from the hardware, which already integrates variables such as image channels and timestamps. In a potential deployment scenario, additional inputs such as visibility and sea state would be provided by the operator. If these variables are not specified, the model's internal mechanism for predicting contextual variables would instead be activated. It follows that future developments will undoubtedly include the improvement of the contextual variable prediction system, which currently represents only a safe fallback in the absence of such variables. The implementation of such a data acquisition system, combined with a more systematic approach to collecting videos or annotated frames during inference, would significantly reduce the resources required to expand the dataset.

In parallel, future work could investigate synthetic data generation techniques to accelerate dataset expansion and improve generalization to rare scenarios. This option has indeed been considered, although concerns remain that reliance on synthetic data might deviate too far from the reality of coast guard acquisitions, leading to datasets closer to the MOB benchmark and thus less representative of operational conditions. Moreover, domain adaptation methods may help bridge the gap between training conditions and real-world deployment, particularly in situations where data are scarce, heterogeneous, or collected across different sensors. The implementation of these strategies, combined with a systematic approach to collecting videos or annotated frames during inference, would significantly reduce the resources required to expand the dataset while enhancing the robustness of the proposed model.

From a development perspective, key improvements include the implementation of multi-GPU support and the development of model export capabilities in ONNX format. The latter in particular would enable integration with existing Leonardo products, which aim to optimize the inference phase and hardware implementation. From a training perspective, the model's generalization capacity should be further enhanced by developing an augmentation strategy that remains consistent with the architecture and the nature of the maritime domain. Careful design of such a strategy is essential to avoid introducing unrealistic variations while still improving robustness across diverse operational conditions. In addition, for both YOLO and MaYOLOc, alternative slicing approaches to SAHI could be investigated. In particular, adaptive tiling strategies, which dynamically adjust the size and placement of image patches according to the content, represent a promising direction. Such methods could reduce redundant computations while preserving small-object detection accuracy, offering a more efficient and context-aware alternative to fixed slicing.

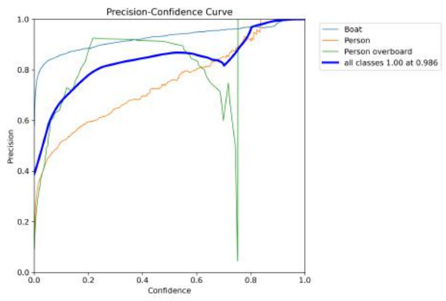
6.2 Final Remarks

In summary, this thesis has demonstrated that one-stage detectors, and YOLO-based architectures in particular, remain the most effective solutions for maritime object detection when real-time performance is required. By proposing MaYOLOc the work has contributed both a methodological advancement and a practical research stimulus. The development of the Coast Guard dataset, together with the experimental evaluation of state-of-the-art detectors, has further consolidated the foundations for continued innovation in this field.

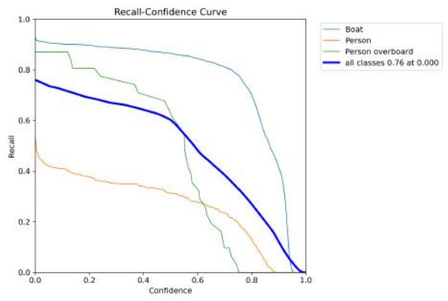
The research has shown that contextual information represents a promising direction for improving object detection performance in complex environments, and that its integration can be achieved without compromising the efficiency of single-stage detectors. While the current implementation still faces limitations, the results confirm the validity of the approach and provide Leonardo with actionable tools for immediate deployment, as well as a roadmap for future improvements.

This work underscores the importance of combining academic research with industrial needs, delivering both a functional solution and a basis for long-term innovation. The contributions made in this thesis are not to be regarded as an endpoint, but rather as the starting point for further developments that will enhance the robustness and applicability of maritime object detection systems in real-world scenarios.

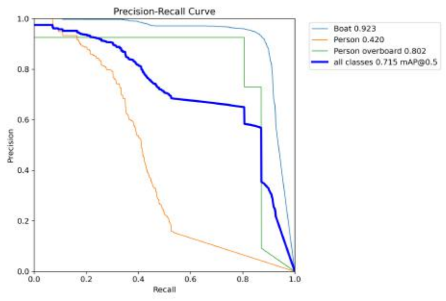
Appendix



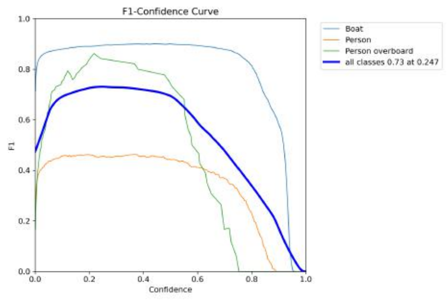
(A) Precision curve



(B) Recall curve

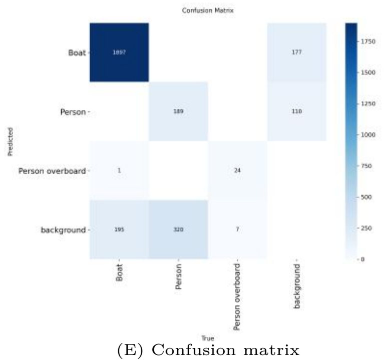


(C) Precision–Recall curve



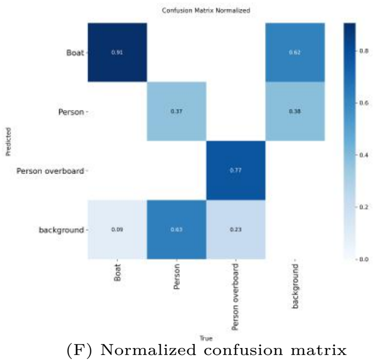
(D) F1 curve

Fig. 1: Precision, Recall, Precision-Recall and F1 curves of YOLOv11s model on CG dataset



	Boat	Person	Person overboard	background
Boat	1897			177
Person		189		110
Person overboard	1		24	
background	195	320	7	

(E) Confusion matrix



	Boat	Person	Person overboard	background
Boat	0.91			0.62
Person		0.37		0.38
Person overboard			0.77	
background	0.09	0.63	0.23	

(F) Normalized confusion matrix

Fig. 2: Confusion matrix and Normalized Confusion matrix of YOLOv11s model on CG dataset

Bibliography

[1] Fatih Cagatay Akyon, Sinan Onur Altinuc, and Alptekin Temizel, *Slicing aided hyper inference and fine-tuning for small object detection*, 2022 IEEE International Conference on Image Processing (ICIP), IEEE, October 2022, p. 966–970.

[2] Lihao Liu et al. Ao Wang, Hui Chen, *Yolov10: Real-time end-to-end object detection*, arXiv preprint arXiv:2405.14458 (2024).

[3] Salvatore Aronica, Francesco Benvegna, Massimo Cossentino, Salvatore Gaglio, Alessio Langiu, Carmelo Lodato, Salvatore Lopes, Umberto Maniscalco, Pierluca Sangiorgi, et al., *An agent-based system for maritime search and rescue operations.*, WOA, vol. 45, Citeseer, 2010, p. 46.

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao, *Yolov4: Optimal speed and accuracy of object detection*, arXiv preprint arXiv:2004.10934 (2020).

[5] Donato Cafarelli, Luca Ciampi, Lucia Vadicamo, Claudio Gennaro, Andrea Berton, Marco Paterni, Chiara Benvenuti, Mirko Passera, and Fabrizio Falchi, *Mobdrone: a drone video dataset for man overboard rescue*, Image Analysis and Processing – ICIAP 2022 (Cham), Springer International Publishing, 2022, pp. 633–644.

[6] ———, *MOBDrone: a large-scale drone-view dataset for man overboard detection*, February 2022.

[7] Zhaowei Cai and Nuno Vasconcelos, *Cascade r-cnn: Delving into high quality object detection*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 6154–6162.

[8] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko, *End-to-end object detection with transformers*, European conference on computer vision, Springer, 2020, pp. 213–229.

[9] Joao Carreira and Andrew Zisserman, *Quo vadis, action recognition? a new model and the kinetics dataset*, proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6299–6308.

[10] Liqiong Chen, Wenxuan Shi, and Dexiang Deng, *Improved yolov3 based on attention mechanism for fast and accurate ship detection in optical remote sensing images*, Remote Sensing **13** (2021), no. 4, 660.

[11] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen, *Up-detr: Unsupervised pre-training for object detection with transformers*, Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 1601–1610.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al., *An image is worth 16x16 words: Transformers for image recognition at scale*, arXiv preprint arXiv:2010.11929 (2020).

[13] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian, *Centernet: Keypoint triplets for object detection*, Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 6569–6578.

[14] Shikha Dubey, Farrukh Olimov, Muhammad Aasim Rafique, and Moongu Jeon, *Improving small objects detection using transformer*, Journal of Visual Communication and Image Representation **89** (2022), 103620.

[15] Yuxin Fang, Bencheng Liao, Xinggang Wang, Jiemin Fang, Jiyang Qi, Rui Wu, Jianwei Niu, and Wenyu Liu, *You only look at one sequence: Rethinking transformer in vision through object detection*, Advances in Neural Information Processing Systems **34** (2021), 26183–26197.

[16] Ross Girshick, *Fast r-cnn*, Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.

[17] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik, *Rich feature hierarchies for accurate object detection and semantic segmentation*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580–587.

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, *Mask r-cnn*, Proceedings of the IEEE international conference on computer vision, 2017, pp. 2961–2969.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, *Spatial pyramid pooling in deep convolutional networks for visual recognition*, IEEE transactions on pattern analysis and machine intelligence **37** (2015), no. 9, 1904–1916.

[20] Glenn Jocher, *YOLOv5 by Ultralytics*, May 2020.

[21] Glenn Jocher, Ayush Chaurasia, and Jing Qiu, *Ultralytics yolov8*, 2023.

[22] Glenn Jocher, Jing Qiu, and Ayush Chaurasia, *Ultralytics YOLO*, January 2023.

[23] Hei Law and Jia Deng, *Cornernet: Detecting objects as paired keypoints*, Proceedings of the European conference on computer vision (ECCV), 2018, pp. 734–750.

[24] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang, *Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection*, Advances in neural information processing systems **33** (2020), 21002–21012.

[25] Zhengbao Li, Jianfeng Dai, Yuanxin Luan, Nan Sun, and Libin Du, *Lr-mpibs: A lora-based maritime position-indicating beacon system*, Applied Sciences **14** (2024), no. 3, 1231.

[26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, *Feature pyramid networks for object detection*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2117–2125.

[27] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, *Focal loss for dense object detection*, Proceedings of the IEEE international conference on computer vision, 2017, pp. 2980–2988.

[28] Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang, *Dab-detr: Dynamic anchor boxes are better queries for detr*, arXiv preprint arXiv:2201.12329 (2022).

[29] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg, *Ssd: Single shot multibox detector*, Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14, Springer, 2016, pp. 21–37.

[30] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang, *Conditional detr for fast training convergence*, Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 3651–3660.

[31] Mahya Nikouei, Bita Baroutian, Shahabedin Nabavi, Fateme Taraghi, Atefe Aghaei, Ayoob Sajedi, and Mohsen Ebrahimi Moghaddam, *Small object detection: A comprehensive survey on challenges, techniques and real-world applications*, arXiv preprint arXiv:2503.20516 (2025).

[32] Goran Oreski, *Yolo* c—adding context improves yolo performance*, Neurocomputing **555** (2023), 126655.

[33] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, *You only look once: Unified, real-time object detection*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.

[34] Joseph Redmon and Ali Farhadi, *Yolo9000: better, faster, stronger*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 7263–7271.

[35] ———, *Yolov3: An incremental improvement*, arXiv preprint arXiv:1804.02767 (2018).

[36] Aref Miri Rekavandi, Lian Xu, Farid Boussaid, Abd-Krim Seghouane, Stephen Hoefs, and Mohammed Bennamoun, *A guide to image-and video-based small object detection using deep learning: Case study of maritime surveillance*, IEEE Transactions on Intelligent Transportation Systems (2025).

[37] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, Advances in neural information processing systems **28** (2015).

[38] Karen Simonyan and Andrew Zisserman, *Very deep convolutional networks for large-scale image recognition*, arXiv preprint arXiv:1409.1556 (2014).

[39] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al., *Gemma 3 technical report*, arXiv preprint arXiv:2503.19786 (2025).

[40] Maxim Tkachenko, Mikhail Malyuk, Andrey Holmanyuk, and Nikolai Liubimov, *Label Studio: Data labeling software*, 2020-2025, Open source software available from https://github.com/HumanSignal/label-studio.

[41] Martin C van Leeuwen, Ella P Fokkinga, Wyke Huizinga, Jan Baan, and Friso G Heslinga, *Toward versatile small object detection with temporal-yolov8*, Sensors **24** (2024), no. 22, 7387.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, *Attention is all you need*, Advances in neural information processing systems **30** (2017).

[43] Wen Wang, Yang Cao, Jing Zhang, and Dacheng Tao, *Fp-detr: Detection transformer advanced by fully pre-training*, International Conference on Learning Representations, 2021.

[44] Yi Wang and Xiang Li, *Ship-detr: A transformer-based model for efficientship detection in complex maritime environments*, IEEE Access **13** (2025), 66031–66039.

[45] Fanyi Xiao and Yong Jae Lee, *Video object detection with an aligned spatial-temporal memory*, Proceedings of the European conference on computer vision (ECCV), 2018, pp. 485–501.

[46] Qianyu Zhou, Xiangtai Li, Lu He, Yibo Yang, Guangliang Cheng, Yunhai Tong, Lizhuang Ma, and Dacheng Tao, *Transvod: End-to-end video object detection with spatial-temporal transformers*, IEEE Transactions on Pattern Analysis and Machine Intelligence **45** (2022), no. 6, 7853–7869.

[47] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai, *Deformable detr: Deformable transformers for end-to-end object detection*, arXiv preprint arXiv:2010.04159 (2020).

[48] Zhiqin Zhu, Renzhong Zheng, Guanqiu Qi, Shuang Li, Yuanyuan Li, and Xinbo Gao, *Small object detection method based on global multi-level perception and dynamic region aggregation*, IEEE Transactions on Circuits and Systems for Video Technology (2024).