

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Exploring Metadata in Neural Networks for UAV Maritime Surveillance

Diogo Samuel Gonçalves Fernandes



Mestrado em Engenharia Informática e Computação

Supervisor: Prof. João Bispo

External Supervisor: Prof. Luís Conde Bento

External Supervisor: Prof. Mónica Figueiredo

July 27, 2023

Exploring Metadata in Neural Networks for UAV Maritime Surveillance

Diogo Samuel Gonçalves Fernandes

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. Pedro Souto

Referee: Prof. Pedro Peixoto

Referee: Prof. João Bispo

July 27, 2023

Resumo

A vigilância marítima ajuda a garantir a segurança e detetar atividades ilegais nos oceanos, sendo feita com o auxílio de embarcações, imagens de satélite e drones, que são cada vez mais usados e utilizam redes neuronais especializadas na deteção de objetos para uma vigilância dos oceanos mais eficaz. No entanto, os drones também apresentam limitações, uma vez que possuem pouco espaço e o consumo de energia é limitado. Além disso, as imagens sofrem com diversos fenómenos que dificultam a deteção de objetos como a espuma produzida pelas ondas, o rasto de barcos e as variações do ângulo e da altitude quando são tiradas.

Ao longo dos anos, diversas soluções apareceram para melhorar a deteção de objetos em ambientes marítimos. No entanto, nenhuma destas abordagens faz uso de outras informações denominadas metadados tal como a altitude, o ângulo da câmara, a hora do dia e as condições atmosféricas com o objetivo de melhorar a precisão da deteção. Estes dados encontram-se muitas vezes disponíveis associados às imagens.

Por este motivo, desenvolveu-se uma rede que é invariante a estas condições de captura de imagens, como altitude e ângulo (outras condições poderiam ter sido consideradas se estivessem disponíveis no dataset utilizado). Para atingir o nosso objetivo integrámos metadados na rede neuronal e utilizámos um método de aprendizagem adversário para que esta pudesse aprender características invariantes de forma a que o modelo não consiga distinguir entre as diferentes altitudes e ângulos de captura das imagens. A nossa arquitetura foi construída sobre o YOLOv7, que é um algoritmo de deteção de objetos em tempo real.

Os resultados revelam que as melhorias alcançadas são pequenas ao fazer com que as redes aprendam características de modo a ficarem invariantes a esses domínios. O *mosaic augmentation* do YOLOv7 foi identificado como uma possível causa uma vez que também aumenta a capacidade do modelo de se tornar invariante a estas diferentes condições. Outra causa possível é o facto de os domínios considerados (altitude e ângulo) não serem ortogonais.

Mais experiências devem ser realizadas através da utilização de datasets que ofereçam metadados mais diversos, assim como condições meteorológicas e marítimas adversas, de forma a serem mais representativas das condições reais e fornecer a oportunidade de definir domínios que sejam independentes entre si. Infelizmente, os datasets disponíveis não têm imagens com estas condições adversas juntamente com metadados, uma vez que incluí-los poderia fornecer mais informações sobre a relevância do nosso modelo.

Abstract

Maritime surveillance helps to ensure safety and monitor illegal activities in oceans, being done with the help of police vessels, satellite images, and **Unmanned Aerial Vehicles (UAVs)**, which are increasingly in use and rely on neural networks specifically designed for object detection tasks to enhance the effectiveness of surveillance operations. Nevertheless, **UAVs** have limitations since space and power consumption are of the essence. Besides that, aerial surveillance images suffer from phenomena like white caps produced by strong winds, boat wakes, glare, and variations in the shooting angle and altitude at which the images are taken.

Over the years, many solutions have been suggested in order to improve object detection in maritime environments. However, none of these approaches uses flight information, such as altitude, camera angle, time of the day, and atmospheric conditions (hereafter referred to as metadata), to improve detection accuracy and network robustness, even though this information is often available and captured by the **UAVs** when doing the surveillance.

This work aims to develop a network unaffected by image-capturing conditions, such as altitude and angle (other conditions could have been considered if provided in the available dataset). To achieve this, metadata was integrated into the neural network, and an adversarial learning training approach was employed. This approach enabled the network to learn invariant features, ensuring that the model could not discern between images based on their altitudes and shooting angles, being built on top of the YOLOv7, which is a state-of-the-art real-time object detector.

To evaluate the effectiveness of this methodology, comprehensive experiments and analyses were conducted. Findings reveal that the improvements achieved by this approach are minimal when trying to create networks that generalize more across these specific domains. The YOLOv7 mosaic augmentation was identified as one potential responsible for this minimal impact because it also enhances the model's ability to become invariant to these image-capturing conditions. Another potential cause is the fact that the domains considered (altitude and angle) are not orthogonal with respect to their impact on captured images.

Further experiments should be conducted using datasets that offer more diverse metadata, such as adverse weather and sea conditions, which may be more representative of real maritime surveillance conditions and provide the opportunity to define orthogonal domains. Unfortunately, current publicly available datasets don't have images with adverse conditions along with metadata, as including them could provide further insights regarding the relevance of this adversarial model training performance.

Keywords: Computer Vision, Remote Sensing, Maritime Surveillance, Domain Adaptation, Adversarial Training, Metadata

ACM Classification: Computing methodologies → Artificial intelligence → Computer vision

Acknowledgements

First and foremost, I would like to thank my supervisors, João Bispo, Luís Conde Bento, and Mónica Figueiredo, for their support, encouragement, and guidance throughout the development of this work. I would also like to thank Professor Ricardo Cruz for the knowledge and resources he provided.

I also emphasize the importance of every teacher who was part of my academic path. In particular, I want to highlight my high school math teacher, who sparked my interest in programming and for helping in my academic and personal development.

To the friends from this course which made this journey more meaningful. The sleepless nights we spent partying or working hard, and the unwavering support we shared, particularly amidst the challenges of online classes during the pandemic.

To my friends from Cabeceiras de Basto who have stood by my side throughout the years, some for over a decade, while others for nearly two. Together, we have shared countless joyful moments and supported each other through challenging times. The amazing conversations and card games at the local coffee shop during afternoons, nights, and early mornings. The endless memories that we created during summer camps and grill parties. Thank you for always being there in all these moments and for your continuous support.

Finally, but not least, I would like to thank my entire family for their presence and support throughout my entire life.

I'm so grateful for having all of you through this phase of my life since this work would not have been the same without your support. It's the end of this journey but the beginning of a bigger one.

Diogo Samuel Fernandes

*“E as pessoas nem sonham que quem acaba uma coisa nunca é aquele que a começou,
mesmo que ambos tenham um nome igual,
que isso só é que se mantém constante,
nada mais”*

José Saramago

Contents

1	Introduction	1
1.1	Context and Motivation	1
1.2	Problem	2
1.3	Goals	2
1.4	Document Structure	2
2	Background	5
2.1	Object Detection	5
2.1.1	Convolutional Neural Networks	6
2.1.2	Algorithms	7
2.1.3	Attention Modules	9
2.1.4	Transformers	11
2.1.5	Metrics	12
2.2	Metadata Integration	15
3	Literature Review	17
3.1	Methodology	17
3.2	Data Gathering in Maritime Surveillance	18
3.2.1	Sensors for Data Gathering	18
3.2.2	Maritime Surveillance Datasets	19
3.2.3	Dataset Comparison	22
3.3	Maritime Object Detection	22
3.3.1	Challenges	22
3.3.2	Data-Level Techniques	23
3.3.3	Network-Level Techniques	24
3.3.4	Optimization of Neural Networks	28
3.4	Summary	29
4	Research Statement	31
4.1	Problem	31
4.2	Hypothesis and Research Questions	32
4.3	Methodology	33
4.4	Summary	33
5	Exploratory Data Analysis	35
5.1	SeaDronesSee Dataset	35
5.2	Data Analysis and Processing	36
5.2.1	Extraneous Variables	36

5.2.2	Relevant Variables	37
5.3	Summary	41
6	Architecture	43
6.1	Model Overview	43
6.1.1	Baseline	43
6.1.2	Proposed Architecture	44
6.2	Implementation Details	46
6.3	Training and Losses	47
6.3.1	First Stage	47
6.3.2	Second Stage	48
6.3.3	Third Stage	49
6.4	Summary	50
7	Results and Discussion	51
7.1	Reproducibility	51
7.2	Baseline and Hyperparameter Tuning	51
7.3	Domain Adversarial Learning	54
7.3.1	Single Domain	54
7.3.2	Multiple Domains using Adversarial Learning	56
7.3.3	Multiple Domains using Model Ensemble	57
7.4	Influence of Mosaic Augmentation	57
7.4.1	Baseline Without Mosaic Augmentation	58
7.4.2	Domain Adversarial Learning Without Mosaic Augmentation	59
7.5	Hypothesis and Research Questions Discussion	61
7.6	Summary	62
8	Conclusion	65
8.1	Conclusions	65
8.2	Contributions	66
8.3	Future Work	67
	References	69
A	Data Annotations	77
A.1	Object Detection Format	77
A.2	Metadata Format	78
B	Hyperparameters	79

List of Figures

2.1	Example of an algorithm spotting objects.	5
2.2	Convolutional Neural Network.	6
2.3	Building block of a Feature Pyramid Network.	7
2.4	Two-stage and one-stage detectors.	8
2.5	YOLOv1 Model Architecture.	9
2.6	Channel and Spatial Attention modules.	9
2.7	Convolutional Block Attention Module.	10
2.8	Self-attention module	10
2.9	Transformer architecture.	11
2.10	Vision Transformer	12
2.11	Area of union and intersection.	13
2.12	Precision-Recall curve.	14
2.13	Integration of metadata into neural networks.	15
2.14	Domain Adaptation using Adversarial Networks.	16
3.1	Sensor types.	18
	(a) Radar.	18
	(b) Infrared.	18
	(c) Optical.	18
3.2	Satellite images.	19
	(a) Radar.	19
	(b) Optical.	19
3.3	Optical images.	20
	(a) Onshore.	20
	(b) Onboard.	20
3.4	UAV optical images.	21
3.5	Real and synthetic images.	21
3.6	Comparison between YOLOv4 and SRC-YOLO.	23
3.7	Implementation of HSF layer.	25
3.8	Convolution Feature Fusion.	26
3.9	PANet Network.	26
3.10	Dilated Attention Module	27
3.11	Ghostnet based architecture.	27
3.12	Comparison between the horizontal bounding box and rotated bounding box.	28
5.1	Distribution of time of the day in images.	36
5.2	Image altitude distribution.	38
5.3	Images captured from UAVs showcasing diverse altitude ranges.	38

(a) 13 meters.	38
(b) 36 meters.	38
(c) 52 meters.	38
5.4 Image angle distribution.	39
5.5 Images captured from UAVs showcasing diverse angle ranges.	39
(a) 8 degrees.	39
(b) 38 degrees.	39
(c) 75 degrees.	39
5.6 Scatterplot between altitude and angle.	40
6.1 High-level diagram of the proposed architecture.	44
6.2 Inference method.	46
6.3 Inference method using ensemble.	46
6.4 Diagram of our architecture with image processing.	47
6.5 First stage of our training loop.	48
6.6 Second stage of our training loop.	48
6.7 Third stage of our training loop.	49
7.1 Evolution of mAP and mAP@50 over the epochs.	53
7.2 Comparison of batches with and without mosaic augmentation.	58

List of Tables

3.1	Publicly available datasets for maritime surveillance.	22
7.1	Performance using default hyperparameters.	52
7.2	Performance using evolved hyperparameters.	52
7.3	Performance using altitude domain.	54
7.4	Performance using the best parameters regarding altitude.	55
7.5	Performance using angle domain.	55
7.6	Performance using the best parameters regarding angle.	55
7.7	Performance using multiple domains.	56
7.8	Performance using the best parameters regarding multiple domains.	56
7.9	Performance using ensemble model.	57
7.10	Baseline performance without mosaic.	58
7.11	Performance on altitude domain without mosaic.	59
7.12	Performance using the best parameters regarding altitude without mosaic.	59
7.13	Performance on angle domain without mosaic.	59
7.14	Performance using the best parameters regarding angle without mosaic.	60
7.15	Performance using multiple domains without mosaic.	60
7.16	Performance using ensemble model.	60
7.17	Average performance of each model.	63
7.18	Average performance of each model without mosaic.	63
B.1	Hyperparameters after evolution over 350 iterations.	79

Abbreviations

- AP** Average Precision. 14
- API** Application Programming Interface. 37
- AR** Average Recall. 14
- AUC** Area Under the Curve. 14
- CA** Coordinate Attention. 27
- CBAM** Convolutional Block Attention Module. 10, 27
- CNN** Convolutional Neural Network. 6, 7, 10, 11, 15, 22–24
- DAM** Dilated Attention Module. 27
- DETR** DEtection TRansformer. 12
- DN** Detection Network. 25
- DNN** Deep Neural Network. 6, 15, 23
- FCN** Fully Connected Neural Network. 15
- FPN** Feature Pyramid Network. 7, 25
- GAN** Generative Adversarial Network. 24
- HOG** Histogram of Oriented Gradients. 5
- HSF** Hierarchical Selective Filtering. 25
- IoU** Intersection over Union. 12–14
- JSON** Javascript Object Notation. 35
- KL** Kullback-Leibler. 49
- MADA** Multi-Adversarial Domain Adaptation. 15
- mAP** Mean Average Precision. 14
- mAR** Mean Average Recall. 14

NMS Non-Maximum Suppression. 6, 8

ROI Region of Interest. 7

RPN Region Proposal Network. 7, 8, 25

SAR Synthetic Aperture Radar. 18, 19, 28

SE Squeeze and Excitation. 10

SURF Speeded-Up Robust Features. 5

UAV Unmanned Aerial Vehicle. iii, 1, 2, 17, 20, 21, 23, 28, 31, 32, 35, 37, 43, 45, 61, 65, 66

ViT Vision Transformer. 11, 12

Chapter 1

Introduction

1.1 Context and Motivation

Remote sensing consists of information acquisition about an object or phenomenon without making physical contact with the object. This technology has a wide range of applications, having received significant attention in recent years [1] in different areas such as wildfire [2] and maritime surveillance [3].

Maritime surveillance regulates and monitors activities that take place in the world's oceans and seas, being done not only by government agencies but also by private companies and organizations. It aims to ensure ships' safety and prevent illegal activities, such as smuggling and piracy, protecting the nautical environment. This can be done with the tracking and identification of ships and vessels, as well as the monitoring of marine traffic and fisheries control.

Data retrieved in maritime surveillance come from different sources, including aircraft. One type of aircraft used for this purpose is the UAV, which is operated remotely or autonomously without a human pilot on board. UAVs are lightweight, have high maneuverability, flexible movement, and have low energy consumption. These aircraft usually have various sensors and embedded devices to collect and analyze data. The data gathered is generally processed in the onboard embedded devices with machine learning and/or deep learning algorithms that can analyze and extract useful information in real-time.

The majority of algorithms used for object detection in maritime environments are extremely complex and require a lot of computational power to be executed. They use image processing and advanced techniques present in computer vision to analyze data collected from the different sensors. However, given that UAVs use embedded systems where the computational capacity is limited, finding solutions that provide good performance while also being computationally efficient is crucial. The designed algorithms for these platforms must balance the accuracy and complexity of the model in order to detect vessels and small ships in real time.

1.2 Problem

UAVs' aerial images differ from those usually used on object detection tasks, which mostly include large objects and a horizontal angle of view. Drone images have different shooting angles, the objects to be detected are often very small, and the features to be extracted are sometimes ambiguous [4]. Images taken over the ocean suffer from other effects, such as extreme glare, boat wakes, and white caps caused by waves [5]. Moreover, adverse weather conditions such as rain and fog will also degrade the quality of images, making it more challenging to perform object detection in this context. Alongside with images, UAVs have other information available on board, such as altitude, camera angle, date, time, and location. This data is commonly referred to as metadata and can offer valuable details.

The embedded devices on UAVs have limited computational power, making it difficult to run some of the larger and more computationally intensive algorithms commonly used for object detection [6]. Making these algorithms achieve real-time inference in edge devices while maintaining an acceptable performance is essential in this application scenario.

This research aims to explore the detection of ships and other objects in the sea using images captured by UAVs and additional flight-related information that may be useful to improve the detection performance while keeping the complexity under control.

1.3 Goals

The primary aim of this work is to tackle the difficulties discussed in Section 1.2, improving maritime surveillance object detection by designing and implementing algorithms that allow the detection of objects using UAVs. Having that in view, the goals of this work are listed as follows:

Development of a neural network to detect objects in a maritime surveillance environment

A neural network will be trained on maritime datasets to recognize objects such as boats, ships, and people. The network will learn the characteristics and patterns present in this environment in order to be able to detect objects in these situations.

Using metadata to enhance the performance of the neural network

Metadata is data that provides information about other data. It can be used to provide context for the image's location, time, and other contextual aspects, providing further insight. By taking into account the metadata of the image, the neural network can potentially make more accurate predictions.

1.4 Document Structure

This document is divided into eight chapters, described as follows:

- Chapter 1, Introduction, briefly describes the context and motivation, as well as the problem and the objectives of this thesis.

- Chapter 2, Background, explores the core concepts regarding object detection and neural networks that are necessary to understand this work.
- Chapter 3, Literature Review, we conduct a review and analysis of state of the art in maritime surveillance.
- Chapter 4, Research Statement, the hypothesis, and the research questions are introduced.
- Chapter 5, Exploratory Data Analysis, the dataset is described, and we perform an exploratory analysis of the relevant data is performed.
- Chapter 6, Architecture, the baseline, and the architecture of the proposed solution are presented, highlighting the key components and innovative design choices that differentiate our approach.
- Chapter 7, Results and Discussion, we discuss the results of our methodology, comparing them with the baseline and providing insights into the performance.
- Chapter 8, Conclusion, presents the principal results and conclusions of this work.

Chapter 2

Background

This chapter aims to introduce the fundamental concepts necessary to understand the work performed in this document. Firstly, Section 2.1 presents a brief introduction to object detection in images, and the algorithms used to perform the detection, focusing on the most used architectures. In the following Section 2.2, we explore the integration of metadata into neural networks to enhance their performance.

2.1 Object Detection

Detecting objects involves analyzing images to detect and identify objects of interest, being this a crucial task in computer vision. Object detection algorithms are responsible for finding the position of the objects and classifying them. The algorithms are trained on large datasets where each one has a bounding box and class associated, as shown in Figure 2.1. The trained model can then be used to recognize objects on other images different than the ones used for training.

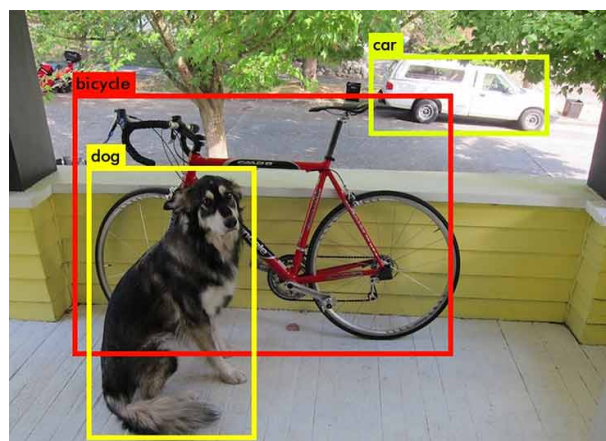


Figure 2.1: Example of an algorithm spotting objects. Extracted from [7].

Classical techniques to extract features such as **Histogram of Oriented Gradients (HOG)** [8] and **Speeded-Up Robust Features (SURF)** [9] were widely used to perform object detection and other computer vision tasks. In recent years, with the many open challenges that still exist in this

area and the advancement of deep learning, object detection has improved significantly by using this new technique [10]. Object detection is a significant area of computer vision, and due to its importance, it will continue to play an essential role in many cutting-edge technologies.

2.1.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs), as seen in Figure 2.2, is a type of **Deep Neural Network (DNN)** used in many computer vision tasks, such as image classification and object detection. This type of neural network consists of multiple layers, where the first set of layers is responsible for extracting the features commonly called the backbone. With the extracted feature maps, the model will make predictions using fully connected layers, also known as the head of the network. Some architectures use a neck as an extra layer connecting the head and backbone to improve and refine the selected features.

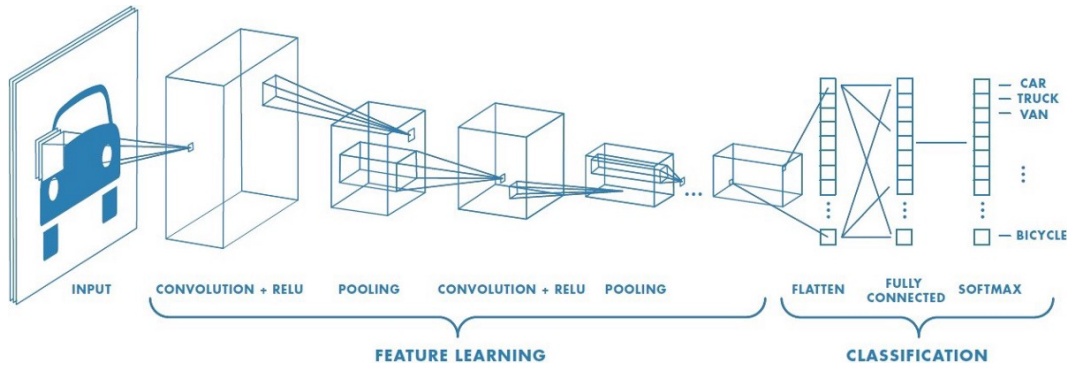


Figure 2.2: Convolutional Neural Network. Extracted from [11].

The network's backbone, also known as the feature extractor, takes an image as input and returns feature maps, employing convolutional and pooling layers. The initial layers are responsible for learning low-level features, whereas the last ones are responsible for learning high-level features. The feature extractor must be able to generalize the extracted characteristics well to make accurate predictions on previously unseen data and be robust to different conditions.

The head receives features at different resolution sizes and predicts a bounding box, and classifies the object. Bounding boxes can be detected using anchor-based or anchor-free approaches. Anchor-based algorithms assign different shapes and sizes to anchor boxes, predefined boxes placed in an image to match and align predicted bounding boxes during object detection. If these shapes and sizes are too specific, the detector head may be unable to detect objects in the image, leading to poor object detection [12]. Since this approach is simple to implement and has shown good performance, it is widely used. On the other hand, anchor-free algorithms directly predict the bounding box without relying on predefined anchor boxes. Instead, they directly estimate the coordinates and dimensions of the bounding boxes based on the features and context of the objects in the image. After bounding boxes are generated, the **Non-Maximum Suppression (NMS)** keeps the bounding boxes with the highest detection scores, deleting the overlapping ones representing

the same item. This algorithm removes bounding boxes with low classification probability scores and then compares the overlap of bounding boxes that belong to the same object.

The neck is an optional component found in some architectures, which helps to improve the overall system's performance by providing additional processing and feature refinement. It makes the connection between the backbone network and the head network. The components that make up the neck network can vary depending on the specific object detection method used. **Feature Pyramid Network (FPN)** is usually employed in the neck and generates a pyramid of features at different scales, which can help to refine the feature maps [13]. **FPN** takes a single-scale image of arbitrary size as input and outputs proportionally sized feature maps at multiple levels. The architecture can be seen in Figure 2.3 and consists of two pathways, a bottom-up pathway corresponding to the backbone and a top-down pathway that increases the resolution of feature maps by upsampling. This structure allows the detection of objects of multiple sizes. Another example of a component used in the neck is the attention modules, which will be introduced in the following sections.

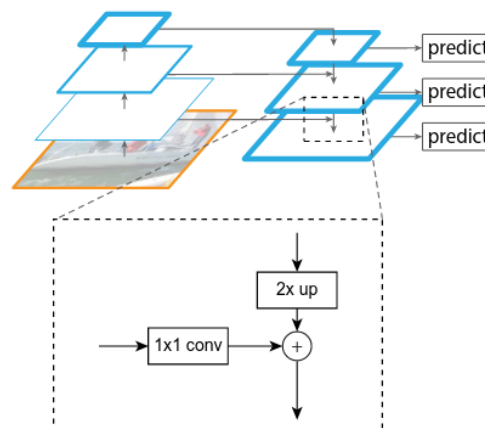


Figure 2.3: Building block of a Feature Pyramid Network. Extracted from [13].

2.1.2 Algorithms

Object detectors based on **CNN** can be divided into two different architectures, as seen in Figure 2.4. Two-stage object detectors are one of the possible approaches to object detection. They use a **Region Proposal Network (RPN)** to generate a set of possible **Region of Interests (ROIs)** in the picture, which are subsequently analyzed to classify each object and refine the position of the bounding box. This architecture has various advantages and can make more accurate predictions but has an increased computational complexity. On the other hand, one-stage object detectors estimate the objects and their locations in an image directly, being more efficient since it does not require the processing of each bounding box separately. However, they may have lower accuracy since they don't use a **RPN** to refine the predictions.

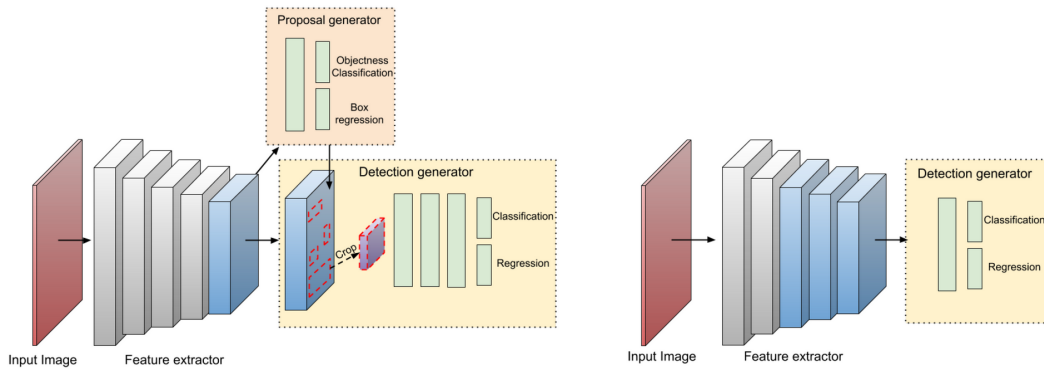


Figure 2.4: Two-stage and one-stage detectors. Extracted from [14].

Two-Stage Detectors Faster R-CNN [15] is a two-stage object detection technique developed in 2015 that is frequently utilized in computer vision applications. It generates candidate regions or proposals that may contain objects using a **RPN** and then classifies and refines the object locations using these proposals. The **RPN** has been taught to recognize items of varied sizes to handle both big and small objects in the images. The **RPN** is used in the first stage of Faster R-CNN to produce proposals by sliding a tiny network across the feature maps of the input image. The **RPN** assigns a score to each proposal based on its likelihood of containing an object and forecasting its position and size. In the second stage, the proposed regions from the previous stage are passed to an object detection network. This network, often an additional branch or module of the model, performs object classification and bounding box regression within each proposed region.

One-Stage Detectors YOLO [7] is an efficient and fast algorithm that can be used for real-time object detection. The algorithm divides the input image into grid cells, where each one predicts a bounding box and a probability score indicating whether or not the grid cell contains an object. Simultaneously, each grid cell predicts the conditional probability of the class to which the object in that grid cell belongs. YOLO predicts a different total of five parameters for each bounding box, four of which determine the box position and the last of which indicates the confidence score. Finally, **NMS** and thresholding are used to remove overlapping bounding boxes and generate the final predictions. This process can be seen in Figure 2.5. Newer versions brought a lot of improvements, such as YOLOv3 [16], which predicts boxes at three different scales and uses short-cut connections that improve the performance for small objects. In 2022, YOLOv7 [17] makes use of a bag of freebies that boosts training time without increasing inference speed, resulting in a better-optimized model. Furthermore, this model employs a new loss function known as focused loss. YOLOv7 uses Extended Efficient Layer Aggregation in the backbone, reducing the size while not affecting the accuracy, and has two different heads. The lead head is in charge of the ultimate output, while the auxiliary head assists with training in the middle layers. YOLOv8 [18] is the most recent architecture, released in 2023. However, it is not well documented, and there isn't a paper about its enhancements regarding previous versions and architecture. Besides YOLO, there are multiple one-stage detectors, such as Single-Shot Detector [19] and RetinaNet [20].

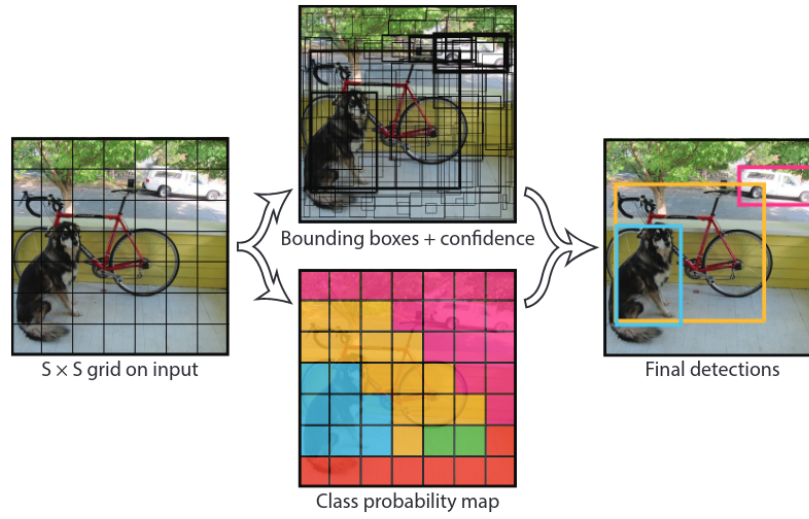


Figure 2.5: YOLOv1 Model Architecture. Extracted from [7].

2.1.3 Attention Modules

Attention mechanisms play a crucial role in the human visual system, aiding in the analysis and comprehension of intricate scenes [21]. Attention modules allow focusing on key information, such as the object, while ignoring unnecessary background information.

Channel Attention focuses on different channels of a multi-channel feature representation, adjusting the importance of each channel. In object detection, most of the time, each image color represents a distinct channel. On the other hand, Spatial Attention focuses on different spatial locations within an image or feature map to dynamically adjust the importance of each location for prediction. Channel and Spatial Attention are the most commonly used forms of attention when detecting images and are illustrated in Figure 2.6.

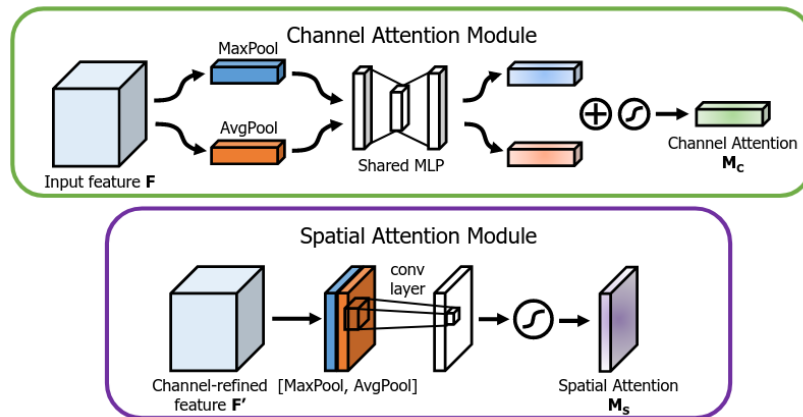


Figure 2.6: Channel and Spatial Attention modules. Extracted from [22].

Squeeze and Excitation (SE) [23] block was one of the first spatial attention modules to appear. The authors designed the **SE** block to improve a neural network's performance by adjusting the different channels' importance. Later, modules that mixed different types of attention surged. The **Convolutional Block Attention Module (CBAM)** [22] is a simple and adaptable module that sequentially infers attention maps along two separate dimensions, channel and spatial, as illustrated in Figure 2.7. The input feature map is multiplied by the attention maps for better feature representation. Any **CNN** architecture can easily use **CBAM** with a minimal performance impact.

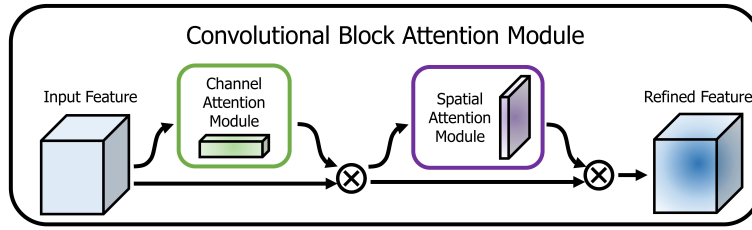


Figure 2.7: Convolutional Block Attention Module. Extracted from [22].

Self-attention is an attention mechanism that has been gaining relevance recently [24]. Each element of the input (x) is transformed into Queries (Q), Keys (K), and Values (V), as can be seen in Figure 2.8. These queries, keys, and values are then used to compute the attention scores, determining how much each input element contributes to the final representation.

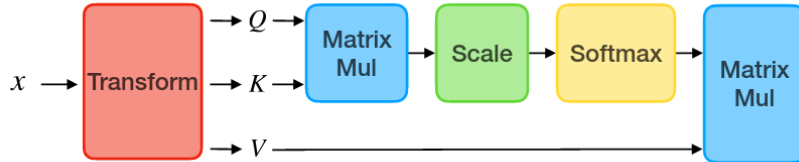


Figure 2.8: Self-attention module. Extracted from [24].

The computation of attention scores (E) is described by equation 2.1 as the dot product of queries (Q) and keys (K), normalized by dividing by the square root of the dimension of the key (D).

$$E = \text{Softmax}\left(\frac{QK^T}{\sqrt{D}}\right) \quad (2.1)$$

Following the calculation of the attention scores, the equation 2.2 describes how the output (O) is produced using the attention scores (E) and values (V).

$$O = \text{Attention}(Q, K, V) = EV \quad (2.2)$$

2.1.4 Transformers

Transformers, as seen in Figure 2.9 are a type of neural network using an encoder-decoder architecture that was introduced in 2017 for natural language processing tasks [25]. The self-attention mechanism, which allows the model to perceive global contexts and long-term interactions, is also a key feature of transformers. Transformers show better scalability than CNNs but need a higher amount of data to train, and if the dataset is too small, the model may not perform as well.

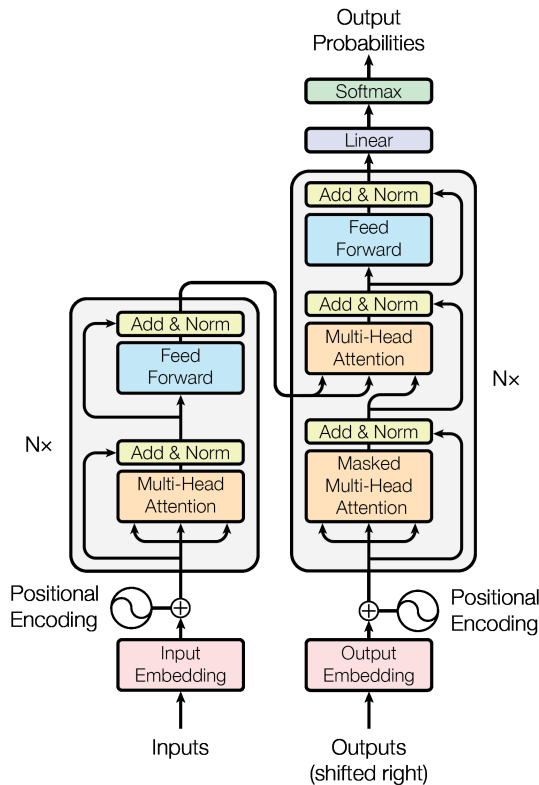


Figure 2.9: Transformer architecture. Extracted from [25].

In recent years, the success of transformers in natural language processing inspired researchers to apply the same principles to computer vision tasks. The result of this work is the **Vision Transformer (ViT)** [26], which uses the transformer architecture for image classification and can be seen in Figure 2.10. All the CNNs in ViT are replaced with transformers resulting in state-of-the-art performance. However, the transformer present in ViT only uses the encoder, disregarding the decoder part. Another difference regarding the transformer is the addition of layer normalization before the sub-layers. The encoder is composed of multi-head attention as well as a multi-layer perceptron. Each head is responsible for capturing different information from the image. For example, one head might focus on learning the textures of the image, while another head might focus on learning the shapes. The final representation of the image is obtained by concatenating the outputs of all heads. First, the ViT converts an image into a sequence of vectors. The input image is divided into a grid of smaller regions or patches. The patches are transformed into patch embeddings through a linear layer and are then processed individually by the transformer

architecture. A learnable class token is added to the patch embeddings, followed by the addition of positional embeddings to obtain the positional information of the image representation. The class token is trained to contain class information and represents the full features, allowing for data classification into different categories. However, for object detection, additional tokens are required that represent the location and size of the objects in the image.

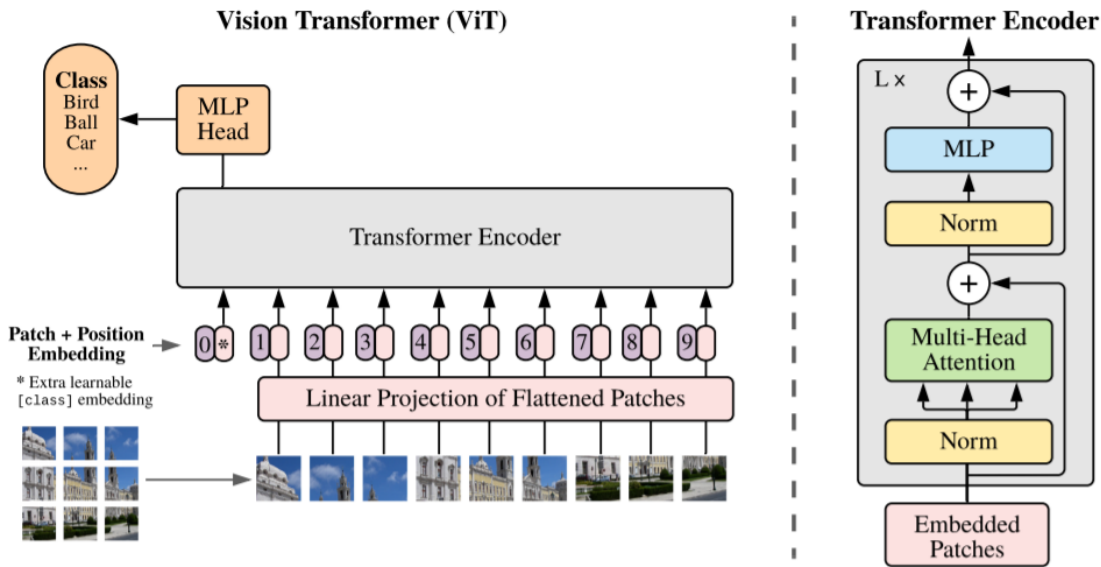


Figure 2.10: Vision Transformer. Extracted from [26].

ViT has faced challenges in terms of computational complexity and image size limitations. One of the solutions to the image size limitation problem is solved by the Swin Transformer [27]. The Swin Transformer replaces the patch-based approach used in ViT with a sliding window mechanism, allowing the model to process images of varying sizes without the need for resizing or padding. This makes it more suitable for real-world applications where images can have different sizes and aspect ratios. DETection TRansformer (DETR) [28] is another transformer-based model that was specifically designed for object detection tasks. DETR uses a combination of a transformer and convolutional layers to process images and directly predict object locations, classes, and confidences from the image.

2.1.5 Metrics

Evaluating the performance of an object detection model is necessary to check if the model is reliable. Due to the different metrics, it is critical to understand them to employ the more suitable one, guaranteeing that the model will provide good results when deployed [29].

Intersection over Union (IoU) is used to evaluate accurately the degree of overlap between the predicted bounding box and the ground truth bounding box. This metric, based on the Jaccard Index [30], is used to determine how similar sample sets are. It measures the intersection ratio

between two geometric shapes, with $IoU \in [0, 1]$, where a value of 0 entails no overlap, and 1 is a perfect match.

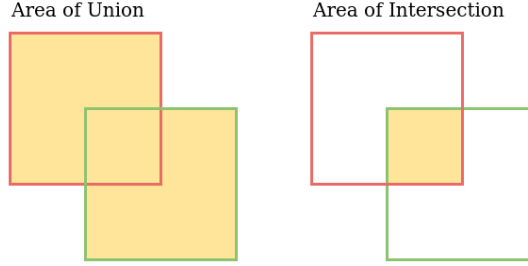


Figure 2.11: Area of union and intersection.

The equation 2.3 refers to the calculation of the IoU , where B_{pd} denotes the prediction's bounding box, and B_{gt} represents the ground-truth bounding box. The area of union and area of intersection is represented in Figure 2.11.

$$IoU = \frac{area(B_{pd} \cap B_{gt})}{area(B_{pd} \cup B_{gt})} = \frac{Area\ of\ Intersection}{Area\ of\ Union} \quad (2.3)$$

The IoU is not used as the direct evaluation metric of the model performance because it provides information about the overlap between the predicted bounding boxes and ground-truth bounding boxes rather than the overall accuracy of the model's predictions. With an IoU and a given threshold α , it is possible to define a True Positive (TP) where $IoU \geq \alpha$ means a correct prediction of bounding box. If $IoU < \alpha$, it will be a False Positive (FP), meaning that the algorithm made an incorrect prediction of the bounding boxes. A False Negative (FN) occurs when the algorithm does not detect the prediction box.

Precision, as shown in equation 2.4, refers to the ability of the model to correctly identify objects of interest among the predicted positive instances. Recall, as shown in equation 2.5, refers to the model's ability to correctly identify all instances of the object of interest in the dataset. For example, if the model detects 100 objects and 90 are correct, the precision is 90 percent. On the other hand, if the model detects correctly 75 objects out of 100 in an image, the recall will be 75 percent.

$$Precision = \frac{TP}{TP + FP} \quad (2.4)$$

$$Recall = \frac{TP}{TP + FN} \quad (2.5)$$

These two metrics are often used together to evaluate the overall performance of an object detection model. Precision-recall curves plot a model's precision and recall values on a graph. It is calculated by selecting the precision values obtained at different recall values.

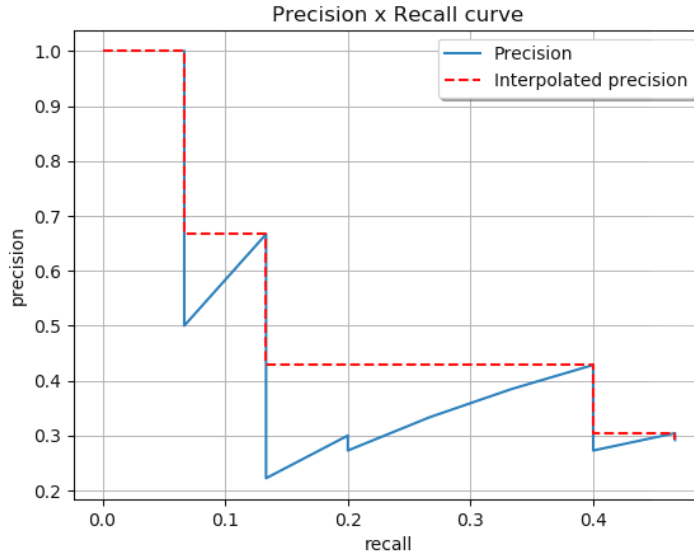


Figure 2.12: Precision-Recall curve. Extracted from [29].

With a precision-recall curve, it is possible to obtain an interpolated precision-recall curve as seen in Figure 2.12. The higher **Area Under the Curve (AUC)** of the interpolated precision-recall curve, the more precision the object detector will have as the recall also increases. The **Average Precision (AP)** of a model is a metric that is calculated with the Area Under the Precision-Recall Curve (AUC-PR). The **Mean Average Precision (mAP)** is the averaged **AP** across all classes [29]. Equation 2.6 is regarding the **AP**, being $p(x)$ the precision where recall is x . Equation 2.7 shows the **mAP** where n is the number of classes and AP_i is the AP in the i th class. Usually these metrics come in the form of **AP@ α** or **mAP@ α** , where α means the threshold. Many algorithms use this notation to evaluate their performance in different datasets such as COCO Dataset [31].

$$AP = \int_0^1 p(x) dx \quad (2.6)$$

$$mAP = \frac{1}{n} \sum_{i=1}^n AP_i \quad (2.7)$$

Besides **AP** and **mAP**, there is also another metric, the **Average Recall (AR)** and **Mean Average Recall (mAR)**. The **AR** is the recall averaged from an **IoU** $\in [0.5, 1]$ and can be calculated as twice the **AUC** of the recall-IoU curve [32]. Equation 2.8 is regarding the **AR**, being $r(x)$ the recall where **IoU** is x and equation 2.9 shows the **mAR** where n is the number of classes and AR_i being the AR in the i th class.

$$AR = 2 \int_{0.5}^1 r(x) dx \quad (2.8)$$

$$mAR = \frac{1}{n} \sum_{i=1}^n AR_i \quad (2.9)$$

2.2 Metadata Integration

DNNs can have different sources of information to train their models. When it comes to computer vision, more information can be attached to images to help the detection, such as metadata. Integrating different types of data, like text and images, can improve their performance, as stated by the work of Sun et al. [33]. In Figure 2.13 it is possible to see a pipeline of their work, where the images are processed through a CNN while the metadata is processed through a Fully Connected Neural Network (FCN). For example, in this case, the data consisted of dermoscopy images, and the metadata contained information regarding the patient's age. The metadata is essential since the probability of a disease is different depending on the patient's age. The outputs from these two networks are merged and fed into another FCN for the final predictions.

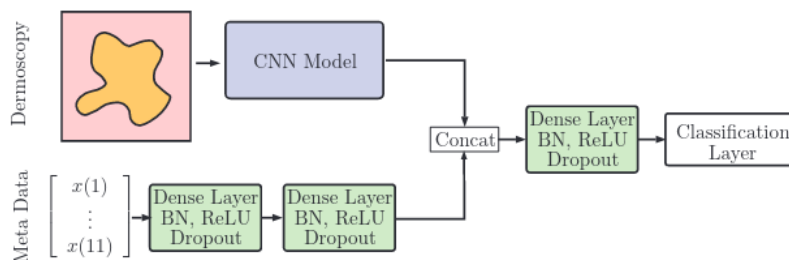


Figure 2.13: Integration of metadata into neural networks. Extracted from [33].

Metadata can also be used to create more generalistic representations instead of merging this data into the neural network to create more accurate predictions, like in the previous case. This is especially useful in datasets where images contain different features and characteristics that make the detection harder, such as weather conditions (i.e., fog, rain, and sunny weather). Each of these conditions can be treated as a distinct domain. Training a model in a specific domain and ensuring its effective generalization to other domains is commonly known as domain adaptation [34].

There are many examples of specialized algorithms that perform domain adaptation, where images trained on a set of domains are evaluated on a completely different domain. Ganim et al. [35] was the first to propose and implement the idea of adversarial domain adaptation. The architecture depicted in Figure 2.14 consists of a feature extractor that processes the input image, followed by a label predictor to classify the image and a domain classifier to determine the image domain. The gradients of the domain classifier are passed through a gradient reversal layer to make the feature extractor learn domain-invariant features. On the other hand, the class labels undergo normal backpropagation so that the network can learn how to classify each class correctly. After this domain adaptation technique using an adversarial network appeared, more works were developed, such as **Multi-Adversarial Domain Adaptation (MADA)** [36], which allowed generalizing to multiple domains instead of just one. By leveraging multiple adversarial domain classifiers, **MADA** offers a more comprehensive approach to domain adaptation, enabling the model to adapt and generalize better across diverse domains.

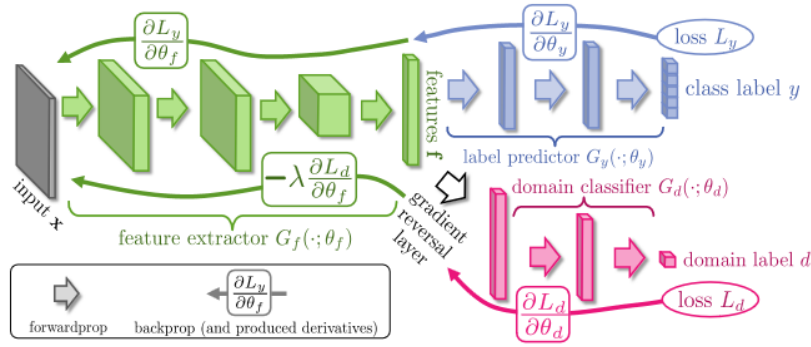


Figure 2.14: Domain Adaptation using Adversarial Networks. Extracted from [35]

The previous approaches are also helpful when we want to make the network invariant to these domains, making the networks more robust. Capozzi et al. [37] created a network invariant to actors present inside a car. Besides that, they employed Deep Bilevel Learning to achieve better generalization, where they gave different weights to each mini-batch, favoring the batches whose gradients match the gradients of the validation mini-batch resulting in better generalization capabilities. When the gradient of the training and validation points in different directions, the weight is either a negative value or zero. Li et al. [38] enhanced the performance of object detection in autonomous driving under adverse weather conditions using adversarial networks. Ferreira et al. [39] applied this approach to create a neural network capable of becoming invariant to signer hand characteristics. Zhenyu Wu et al. [40] employed these adversarial networks to enhance the model's ability to generalize effectively across diverse domains in drone images, which is a similar case to the one addressed in this thesis. This approach ensured that the underlying network focused on extracting only relevant features by becoming invariant to three distinct domains, which were altitude, angle, and weather.

Besides adversarial learning techniques, there are other methods to perform this task. To learn invariant features across different domains, Tzeng et al. [41] proposed a methodology that minimizes the distance between source and target representations of two distinct networks.

Chapter 3

Literature Review

This chapter provides a review of the current state of the art regarding maritime surveillance. Our focus is to analyze the techniques that have been employed to perform effective maritime object detection. First, in Section 3.1, we provide a comprehensive overview of the methodology that will be employed for conducting the literature review. The literature review starts with an analysis of publicly available datasets, which are presented in Section 3.2. Furthermore, in Section 3.3, we will analyze and describe state-of-the-art approaches to perform maritime object detection. Finally, optimization methods employed by researchers to improve the performance of neural networks are outlined.

3.1 Methodology

Our literature review process was guided by specific research questions that align with the objectives of this dissertation. We formulate the following literature research questions (LRQs) to guide our literature review:

LRQ1. Which datasets are used to train models in maritime surveillance?

The performance of object detection models in maritime surveillance heavily relies on the quality and type of data used for training. Different data types, such as radar, infrared, or color images, require different algorithms to extract relevant information effectively. Therefore, it is essential to have a deep understanding of the different types of data available in this domain to choose the best algorithms to solve our problem.

LRQ2. Which algorithms are used for object detection in maritime surveillance?

Creating reliable algorithms for object detection using UAVs in maritime environments is one of the main goals of this work. Considering this, it is crucial to understand and know what was done in this field to create reliable models and choose an appropriate baseline based on the existing models.

LRQ3. How to improve the performance of object detection algorithms in edge devices?

We need to ensure that our object detection algorithm is capable of running on an edge device. Therefore, it is essential to have an understanding of the optimizations performed by researchers on neural networks for edge devices in order to identify opportunities for further optimization.

3.2 Data Gathering in Maritime Surveillance

In supervised learning, data is necessary to train and evaluate models. Its quality, quantity, and diversity have a significant impact on the model's performance. Thus, it is important to understand the data acquisition and manipulation processes required to build a dataset. These datasets can be used to improve the performance of object detection algorithms and can lead to new advancements in the field.

3.2.1 Sensors for Data Gathering

The most widely used sensors to gather data in maritime surveillance are radar sensors, as well as infrared and optical cameras [42]. These sensors are commonly used on satellites, aircraft, ships, and shore regions, each one having its unique capabilities and limitations. Figure 3.1 shows images captured with three different sensors.

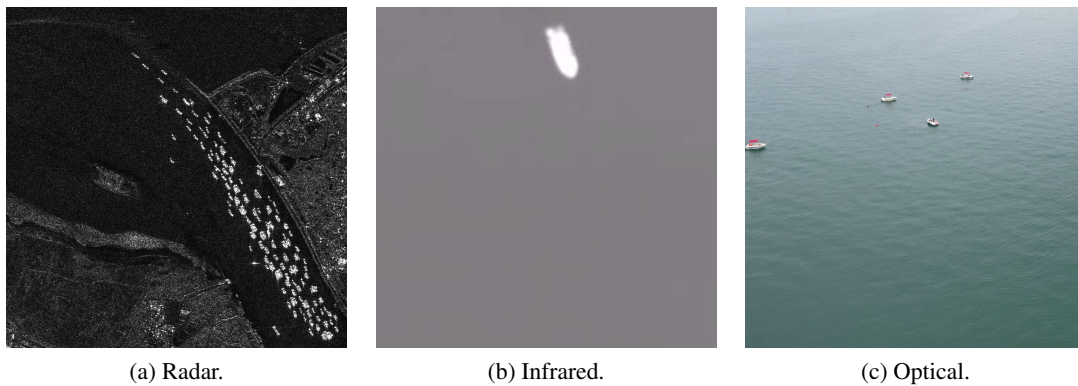


Figure 3.1: Sensor types. Extracted from [43], [5] and [44]

Radar sensors are used to capture **Synthetic Aperture Radar (SAR)** images. This kind of image can easily detect objects despite the weather conditions [45]. However, radar images cannot often detect small objects since they produce weak radar signals [46]. Due to this poor small object detection, they cannot detect small wooden boats and rafts in the sea [47] and are more costly compared to other sensor options.

Thermal infrared cameras work by detecting the heat emitted by objects rather than relying on the object's light emission. This makes them particularly useful for nighttime applications, as they do not require any external illumination to produce an image [46]. Nevertheless, the images

of these sensors are black and white, so there is limited color information available for posterior analysis. Infrared cameras also have lower image quality than optical cameras.

Optical images can accurately detect ships from all ranges when used with image classification techniques. Optical images can be stored for a more thorough analysis, allowing humans to review later images of detected objects. However, these images are affected by poor weather conditions such as rain, waves, and glare from the sun's reflection on the water [5], which need to be handled appropriately.

3.2.2 Maritime Surveillance Datasets

Datasets collected with different sensors and with different devices are used for different purposes. Public maritime surveillance datasets are mainly used for detecting maritime objects and humans in the water. We will examine the characteristics of datasets available in the literature and how they can be used to train and evaluate machine learning models for maritime surveillance. The datasets that will be analyzed in the following sections are listed in Table 3.1.

3.2.2.1 Satellite Images

Satellites are often equipped with radar sensors. SAR Ship Detection Dataset [48] is a dataset used for ship detection on synthetic aperture radar images, having three different shapes of bounding boxes and a total of 1160 images, including 2551 labeled ships. High-Resolution SAR Images Dataset (HRSID) [43] is another dataset with SAR images built for ship detection and instance segmentation with a total of 5604 images and 16951 ships. In addition to radar sensors, optical cameras are employed on satellites to capture images and gather visual data. Airbus Ship Detection Dataset [49] is a dataset of satellite optical images which contains more than 100,000 images. The difference between these types of images is illustrated in Figure 3.2.

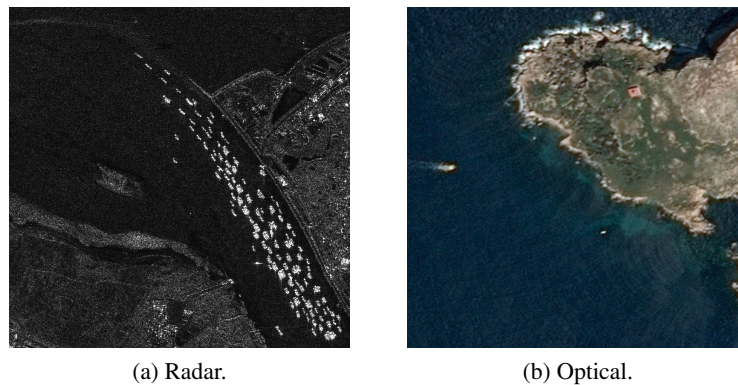


Figure 3.2: Satellite images. Extracted from [43] and [49].

3.2.2.2 Onboard and Onshore Images

Images obtained onboard and with shore cameras provide a different perspective close to the surface, as shown in Figure 3.3. SeaShips [50] is a shore-based dataset composed of 31455 images captured from video segments from monitoring cameras. These cameras are deployed on the coastline of China. This dataset has six different categories of ships to be detected, containing different scales and illumination conditions. Similar to the last one, KOLOMVERSE [51] is a dataset with five different classes for maritime object detection, but captured onboard in South Korea. This dataset englobes different visibility conditions and illumination and has approximately 2 million images. Singapore Maritime Dataset (SMD) [52] is another dataset composed of shore and onboard annotated videos. This dataset has been captured at different times of the day, such as sunrise, afternoon, evening, and sunset. The Water Surface Object Detection Dataset (WSODD) [53] is a collection of 7,467 images taken in different environments, climate conditions, and shooting times. The dataset also includes 14 different object categories, having a total of 21,911 instances, and is not limited to detecting ships, having other objects to be detected, such as buoys and rocks.

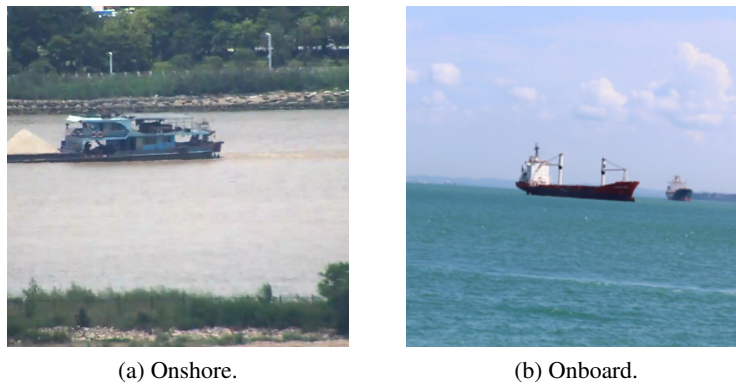


Figure 3.3: Optical images. Extracted from [50] and [52].

3.2.2.3 UAV Images

Figure 3.4 shows some images illustrating what can be captured with optical cameras in drones. We can see that UAV images pose challenges in image-based surveillance since they have a variable angle perspective and altitude, significantly impacting the size of objects to be detected.

In 2019, Ribeiro et al. [5] presented a dataset containing videos from UAVs captured on the coast of Lisbon. The videos are captured at different altitudes and shooting angles, comprising a different range of conditions such as glare, boat wakes, and waves that make the detection of ships in this environment more challenging. Chen et al. [54] also created a dataset with videos taken under different weather conditions, angles, and altitudes. It includes images captured in a foggy environment with many small ships, which can be challenging to detect. MOBdrone [55] is another dataset with more than 125 thousand drone images, annotated from videos. The dataset

has five types of objects to detect in images with an altitude ranging from 10 to 60 meters. These images were also captured under different weather conditions in Pisa, Italy. However, neither of these datasets includes flight-related information, such as altitude and shooting angles.

SeaDronesSee [44] is a recent dataset published in 2022. The authors collected data and annotated over 54,000 frames from a high set of altitudes and viewing angles from the videos. They evaluated state-of-the-art algorithms on their dataset and provided a baseline. This dataset also has metadata associated with the present images.



Figure 3.4: UAV optical images. Extracted from [5] and [44].

3.2.2.4 Synthetic Datasets

The difficulty of gathering real-world data associated with the cost, and legal limitations of using UAVs in these areas, led to the creation of synthetic datasets, as pointed out by Kiefer et al. in their study [56]. They created a synthetic dataset that includes a variety of objects with metadata annotations. This allowed the training and testing of object detection algorithms without needing UAVs or potential legal limitations. However, synthetic datasets frequently lack realism in their images and models. This can cause that trained algorithms under synthetic datasets may not be able to generalize very well. A comparison between real and synthetic datasets can be seen in Figure 3.5.

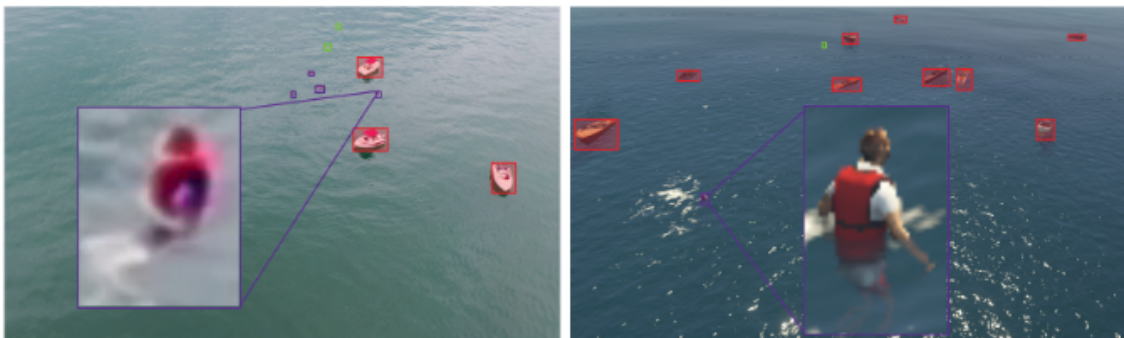


Figure 3.5: Real (left) and synthetic (right) images. Extracted from [56].

3.2.3 Dataset Comparison

Table 3.1 contains a brief summary of the datasets described in this section. In maritime surveillance, various types of sensors are used to collect data. However, optical ones are more popular due to the benefits mentioned previously. It is also worth noting that only two datasets in this area provide image metadata: SeaDronesSee [44], and the DGTA-SeaDronesSee [56].

Table 3.1: Publicly available datasets for maritime surveillance.

Dataset	Platform	Sensor	Media Type	Objective	Metadata
HRSID [43]	Satellite	Radar	Image	Ships	No
SSDD [48]	Satellite	Radar	Image	Ships	No
Airbus Dataset [49]	Satellite	Optical	Image	Ships	No
SMD [52]	On-shore	Optical	Video	Ships	No
	On-board	Near-infrared			
Seaships [50]	On-shore	Optical	Image	Ships	No
KOLOMVERSE [51]	On-board	Optical	Image	Maritime Objects	No
WSODD [53]	-	Optical	Image	Maritime Objects	No
Seagull [5]	UAV	Optical Infrared Hyperspectral	Video	Ships	No
Chen et al. [54]	UAV	Optical	Video	Ships	No
MOBDrone [55]	UAV	Optical	Video	People	No
SeaDronesSee [44]	UAV	Optical	Image	Ships People	Yes
DGTA- SeaDronesSee [56]	Synthetic	Optical	Image	Ships People	Yes

3.3 Maritime Object Detection

Different approaches have been taken over the years in order to improve maritime object detection. Despite the progress made, there are still many challenges and open questions that need to be addressed in order to get the best possible solution for this problem. First, we will analyze the most typical approaches for detecting maritime objects in images, and then we will briefly describe other techniques used, such as object detection in video.

3.3.1 Challenges

Adverse Conditions Cruz and Bernardino [57] proposed a simple CNN model to detect small ships on the ocean training with the Seagull dataset. The model was first pre-trained on a large dataset and then fine-tuned to be able to detect maritime objects. When seeing the results of their proposed method, using a precision-recall curve, they found out that there is a huge drop in performance beyond 80% recall. This decrease shows the presence of harsher conditions, such as intense glare and white caps caused by waves making it harder to classify objects.

Multi-Scale and Small Objects Lee et al. [58] compared the performance of two different CNN models on the SMD dataset. The architectures were similar, where one of them had a passthrough layer in order to enhance the performance for small object detection. Passthrough layers pass their inputs directly to their outputs without performing any calculations. These layers are used to preserve the spatial information of the feature map as it passes through the network. The results led to the conclusion that small representations of ships are vanishing in deeper layers which results in a worse performance.

Real-time Detection Liu et al. [59] stated that the limited energy consumption and computing power of embedded devices used in UAVs restrict the applications of some DNN that perform object detection. For this reason, the deployed algorithms on embedded devices need to be efficient. This is important because these models need to perform object detection in real-time.

3.3.2 Data-Level Techniques

Improving object detection requires not just changes to deep learning models but also attention to the training data. Preprocessing and augmenting it properly can significantly enhance object detection network performance.

3.3.2.1 Image Restoration

As previously mentioned, harsh conditions such as intense glare and waves present a significant problem in object classification. These conditions are usually addressed by preprocessing and restoring images to minimize the effects of fog and other factors that can negatively impact image quality. SRC-YOLO [60], a neural network based on YOLOv4-tiny, aimed to tackle the issue of foggy images by incorporating Single Scale Retinex [61] defogging algorithm at the input level. As seen in Figure 3.6, this technique improves image clarity by reducing fog.

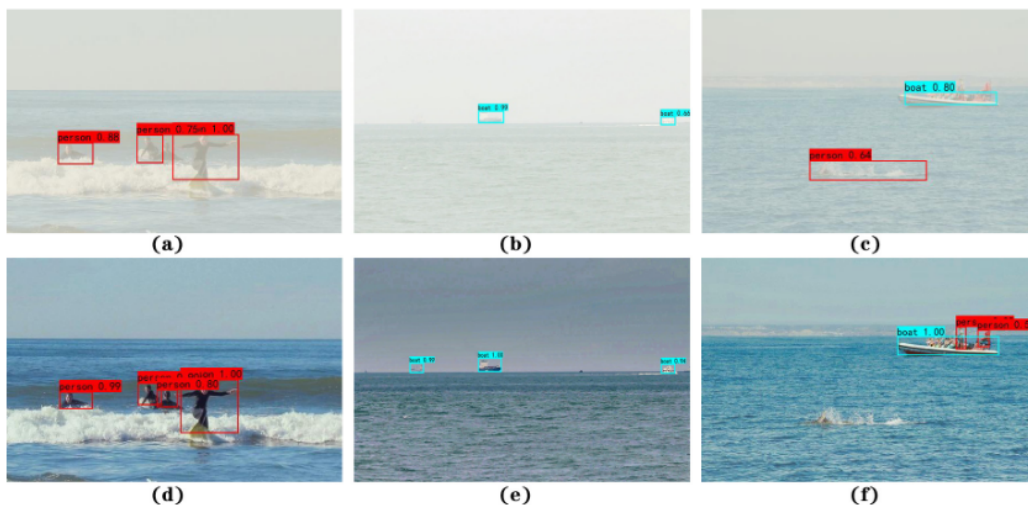


Figure 3.6: Comparison between YOLOv4 and SRC-YOLO. Extracted from [60].

However, Nie et al. [62] stated that image restoration techniques could potentially harm the detection of objects because these techniques could degrade existing image characteristics, making object detection more difficult. For this, in their research, they synthesized hazy and low-light images. The researchers trained a neural network using three approaches: without synthetic images and image preprocessing, with image preprocessing only, and using only synthetic images. They discovered that the third approach, utilizing synthetic images in training, resulted in the best performance. However, the first two approaches, which involved image preprocessing, produced nearly identical results. They further stated that this phenomenon should be better studied because it could be particular to the dataset. However, the experimental results were evident, showing that haze weather and low illumination affected the ship detection accuracy and that models trained with those images were more robust.

Other authors started training models using synthetic data to train their networks to improve their performance. Liu et al. [63] also generated synthetic data bearing in mind the different weather conditions. They evaluated the model on the Seaships dataset under adverse imaging conditions, and it was found to outperform other representative methods. The study also identified areas for future improvement, such as using transfer learning and **Generative Adversarial Networks (GANs)** to produce more natural-looking synthetic images.

Although synthetic data can be used mainly to simulate bad weather conditions, it can be helpful in other situations. A small number of small ships on images can lead to poor detection by the algorithm due to limited sample size, as noted by Z. Chen et al. [64]. The authors propose a hybrid deep learning method to improve small ship detection, combining a modified **GAN** with a **CNN** based detection method. After training with synthetic data, the performance in detecting small ships significantly improved.

3.3.2.2 Image Tiling

In 2023, Zhao et al. [65] proposed a new object detector based on the YOLOv7. The authors acknowledge that enlarged images make the small objects more pronounced but require more memory. To achieve better performance and extract more information on the SeaDronesSee dataset without increasing memory usage, the researchers cropped the image into four patches. They then trained on these cropped images, which saved memory and enlarged the image size. This method is similar to Slicing-Aided Hyper Inference [66], which divides the input images into overlapping patches at inference time, leading to an improvement in the detection due to larger pixel areas for small objects. This technique is often applied to optical satellite [67] and drone [68] images.

3.3.3 Network-Level Techniques

Small objects, intense glare, and boat wakes are the hardest challenges for object detection in maritime surveillance. The performance of object detection systems can be significantly improved by implementing some specialized techniques. For example, making predictions at different scales is valuable when dealing with images of maritime vessels captured from diverse perspectives and

altitudes. If the object detector is trained solely on one scale, it may struggle accurately to detect vessels in images with different sizes. The object detection system can also benefit from the use of other techniques, such as attention mechanisms, which contribute to an overall improvement in performance and reliability. HSF-Net [69] is a network that embeds feature maps in different scales to the same space using a **Hierarchical Selective Filtering (HSF)** layer, as can be seen in Figure 3.7. The network is similar to the Faster R-CNN, having a feature extraction network with multiple convolutional and pooling layers, a **RPN** and a **Detection Network (DN)**. The **HSF** layer is applied both to **RPN** and **DN**. This layer works by focusing on extracting the shape instead of detailed local features. Three scales are utilized to embed objects of varying sizes into the same dimensional feature space. The embedded feature maps are then transformed into fixed-sized vectors, which are fed into two fully connected layers, generating object proposals and performing bounding box regression. Chen et al. [45] proposed and tested a similar method based on an optimized **FPN**. The **FPN** was embedded in a traditional **RPN** and then mapped into a new feature space. The proposed network outperformed previous models in complex scenarios, showing that it is appropriate both for multi-scale and multi-target recognition as well as the detection of small ships.

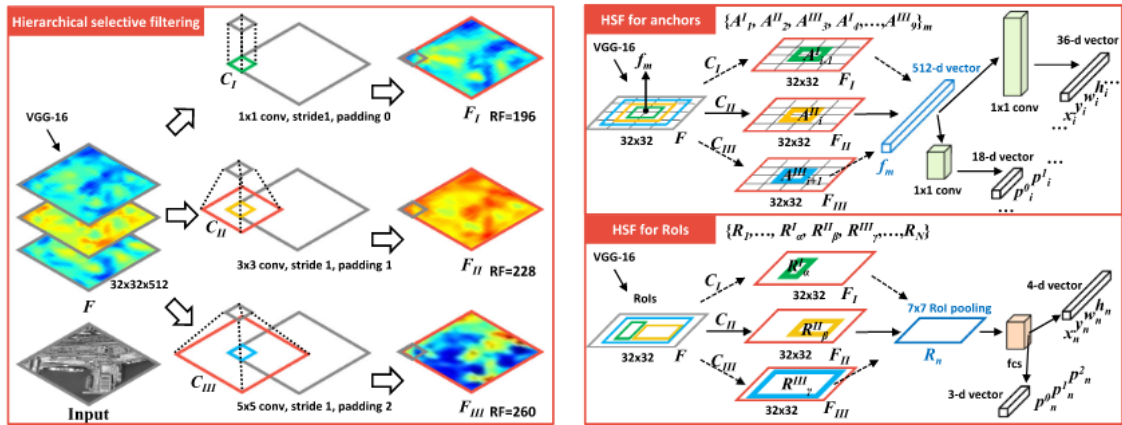


Figure 3.7: Implementation of HSF layer. Extracted from [69].

These approaches can also be employed on one-stage detectors, resulting in better performance and faster processing. Zhang et al. [70] created a multi-layer convolution feature fusion (CFF-SDN), as can be seen in Figure 3.8. They used convolutions with residual connections to extract features at different levels. The CFF-SDN model fuses shallow and deep feature maps, improving ship target detection at different sizes. It uses multiple fused feature maps for classification and regression and employs pruning to increase detection speed. VarifocalNet* [47] improves accuracy and reliability in harsh sea conditions using other techniques besides multi-scale prediction such as deformable convolution, redesigned loss function, soft non-maximum suppression. Results show that the proposed model outperforms similar methods in terms of accuracy and robustness for small objects. This model is specifically designed to enhance object detection in harsh maritime environments.

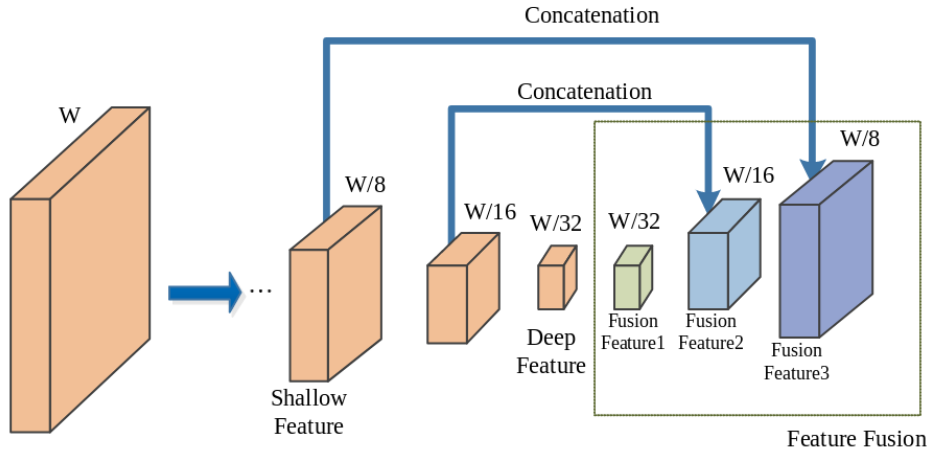


Figure 3.8: Convolution Feature Fusion. Extracted from [70].

YOLOv3 was also enhanced by Liu et al. [71], who developed a cross PANet, as can be seen in Figure 3.9. This is a feature fusion structure for cross-feature maps. In addition to multi-scale features, properly adjusted anchor boxes are crucial to ensure that no objects are missed. For this, they proposed two different anchor box settings, namely the average method and the select-all method. Hong et al. [72] also improved the anchor boxes on YOLOv3 by using a linear scaling based on the k-means++ instead of the k-means clustering algorithm. They also used a Gaussian model to output the uncertainty of each prediction bounding box improving the detection accuracy. Four anchors are assigned in the detection layer to increase the robustness of this model for detecting objects in multiple scales. This improvement resulted in a better performance in the detection of maritime objects from both radar and optical images from satellites.

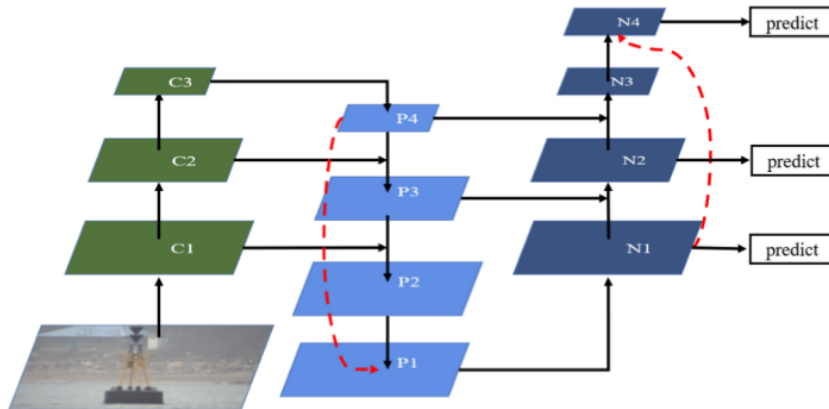


Figure 3.9: PANet Network. Extracted from [71].

Researchers in this field are also using attention modules to improve their networks. Li et al. [73] proposed an improved YOLOv3 tiny network. The authors used convolution layers instead of max-pooling and expanded the number of input channels for prediction, improving the detection of the small target detection. They also added one attention module at the end of the

backbone, the **CBAM**. Chen et al. [74] proposed **Dilated Attention Module (DAM)** to extract feature representations of ship targets on YOLOv3. This lightweight attention module has a larger receptive field that can capture a wider variety of surrounding information, which helps discern small ship targets in harsher environments. As depicted in Figure 3.10, the **DAM** is very similar to the **CBAM** module, having channel attention followed by spatial attention. It is important to note the existing residual connection present in this module, which allows small and vanishing features are not lost, and the dilated block is composed of dilated convolutions.

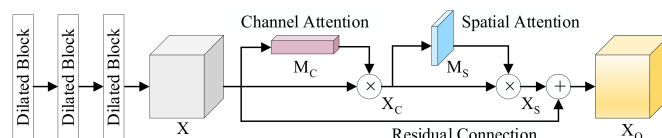


Figure 3.10: Dilated Attention Module. Extracted from [74].

RetinaMFANet [75] is another model that is improved using an attention module, having reliable ship detection in low-illumination environments. The authors also proposed a new multi-scale feature fusion network structure. Li et al. [76] use the **Coordinate Attention (CA)** module, putting it on the last layer of the backbone of the YOLOv5. The **CA** embeds positional information into channel attention, allowing the network to focus on large regions without introducing a large amount of processing. Later, they proposed new architecture [77], which had a backbone with both convolutions and transformers, as can be seen in Figure 3.11. In the neck, the feature maps are fused based on GhostNet, and the head employs three detectors of different scales to calculate the position and size of the objects. The use of two GhostNets allowed the optimization of the network. Full transformer networks are already starting to appear, even though they have bigger complexity than the networks based on convolutions. Sun et al. [6] uses the Swin Transformer, which is a hierarchical transformer. The Swin Transformer is the backbone network for Faster R-CNN, a two-stage detector. The main disadvantage of this method is the computational cost.

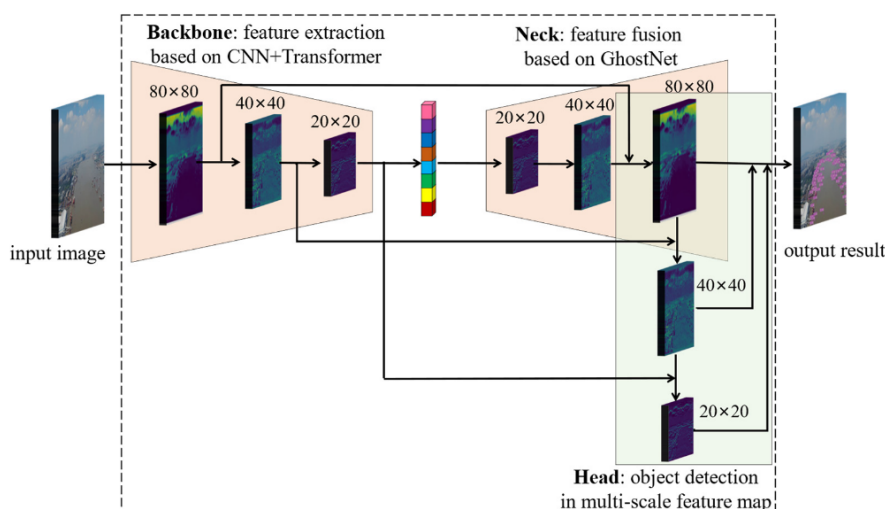


Figure 3.11: Ghostnet based architecture. Extracted from [77].

Due to the challenges associated with detecting small objects, researchers are actively exploring innovative approaches. Duan et al. [78] have attempted to detect small objects in maritime environments using eagle-eye mechanisms. Eagle-eye mechanisms refer to techniques such as glare reduction, dim control, color enhancements, and background adaptation.

In 2018, Yang et al. [79] proposed a new way to detect rotated ships from aerial images and a year later proposed an enhanced version [80]. In 2022, Shao et al. [81] also proposed a rotated box detection for ships, but this time specific for SAR imagery. The objective of these efforts was to minimize overlapping areas. Their work is more relevant for detecting large clusters of ships docked in the port, as seen in Figure 3.12. However, using rotated boxes to detect ships may include some computational overhead, which we cannot handle when considering real-time ship detection.

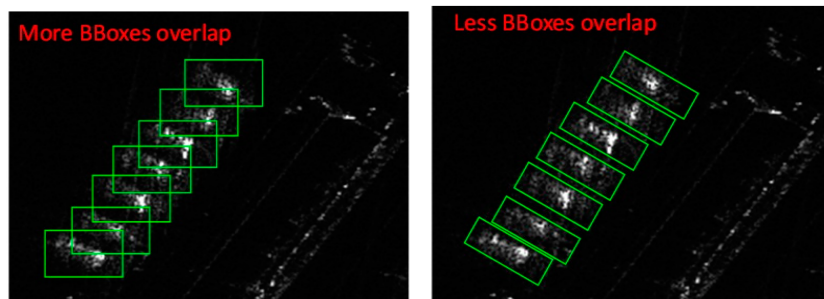


Figure 3.12: Comparison between the horizontal bounding box and rotated bounding box. Extracted from [81].

Videos, as opposed to static images, capture visual information and the temporal dynamics of an event. Extracting temporal features is one of the biggest benefits of using videos. Using spatial-temporal data in image detection can improve performance by providing additional context about the ships and their movements within the image. It can also help to reduce false positives by giving temporal cues for object tracking. Cruz et al. [82] analyzed the spatial-temporal data on videos from the Seagull dataset with challenging conditions such as small ships and different vessels, adverse weather, different altitudes, and backgrounds. The proposed method had one spatial feature extractor based on the VGG network and one recurrent layer, the long short-term memory. Their work shows that learning temporal features is helpful for maritime detection in videos captured by UAVs. Chen et al. [54] also analyzed spatial-temporal data of successive frames from their proposed video dataset based on kinematic ship information.

3.3.4 Optimization of Neural Networks

Some optimization techniques were already used by researchers in the context of maritime surveillance to make the networks faster and smaller. These techniques are fundamental in order to make the algorithms more efficient and able to perform real-time detection in embedded systems.

Liu et al. [83] added a depthwise separable convolution on the backbone and the feature fusion of a YOLOv4. With this, they reduced the number of weights by more than 40%, improving its efficiency. Yang et al. [84] proposed a lightweight network using a ShuffleNet and YOLOv5, which also employs a feature fusion to combine different feature maps. Nalamati et al. [85] compared two networks, the CenterNet and EfficientDet, to find the most efficient one. They tested both architectures on SMD and SeaShips datasets. The results show that CenterNet is a good choice for real-time detection systems and provides good accuracy and speed. On the other hand, EfficientDet models overall had better accuracy than CenterNet but lower speed. The authors suggest that CenterNet model is a good choice for real-time detection systems, while EfficientDet models are better for offline analysis of captured surveillance videos.

Sharafaldean et al. [86] collected a maritime dataset and used the YOLOv4 algorithm to perform the object detection. They deployed the algorithm on a Jetson Xavier NX, stating that a possible way to optimize the algorithm would be with network pruning and quantization techniques. Zhang et al. [70] and Xu et al. [87] pruned the channels reducing the inference time. On the other hand, Liu et al. [59], to compress the width and depth of the model, performed channel and layer pruning simultaneously.

3.4 Summary

In Section 3.1, we start this chapter by outlining the three distinct literature research questions. These LRQs guided our literature review, ensuring that we addressed specific and relevant aspects within our research domain. Section 3.2 analyzed in detail the different datasets available for this task regarding images from different sensors, finding an answer to the LRQ1 about which datasets were used to train models in maritime surveillance. Regarding the LRQ2, the majority of algorithms to detect maritime objects in literature are very recent, meaning that the interest of researchers in maritime surveillance is increasing. These algorithms are detailed in Section 3.3. Many distinct strategies to improve the detection of objects in maritime environments can be found, addressing the problems of small object detection and image quality issues. Different methods are employed, such as multi-scale learning, changing the anchor boxes, loss functions, and using attention modules. Image augmentation and preprocessing are also frequently used to enhance their quality. Since the vast majority of solutions presented in literature use large models and very complex architectures, optimization of these networks was outlined in Section 3.3.4 to answer the LRQ3. Using lightweight neural networks such as depthwise separable convolutions and techniques like network pruning can improve the efficiency of object detection algorithms. Despite this, the performance of neural networks is not a priority for many researchers, who instead focus on achieving the best performance possible, even if this means using less efficient models.

Chapter 4

Research Statement

This chapter provides an overview of the problem this work aims to solve. In Section 4.1, we begin by presenting the problem, taking into consideration the gaps that exist in the current literature. After the problem is presented, we formulated the hypothesis and research questions in Section 4.2. Finally, in Section 4.3, we describe the experimental methodology that we are going to use to validate our work.

4.1 Problem

We identified several gaps in the existing literature on maritime surveillance and uncovered promising avenues for further investigation in this field. UAVs are widely recognized for capturing images from various altitudes and angles, subject to external factors like weather conditions and lighting variations. While these variations may not directly impact object recognition, they can pose challenges for detection. Using the metadata presents an opportunity to enhance existing models and potentially improve algorithms in this specific domain, as stated by some researchers [82]. Researchers have conducted investigations regarding the use of metadata, and they have obtained promising results when applying this approach to UAV images captured in terrestrial environments [40]. However, when it comes to maritime surveillance, there is a lack of studies exploring the utilization of metadata, although there are already datasets containing this information. Furthermore, utilizing this information without compromising network performance is something to strive for to enhance the networks.

Another promising direction for enhancing object detection in maritime surveillance is exploring and leveraging the temporal correlation between images. However, it often involves using more complex networks, which is not our primary objective.

To build upon these findings, we explored a way to integrate metadata in a neural network, as it had not been previously done in the context of maritime images. Our objective is to develop a network that disregards irrelevant factors in the images and emphasizes the relevant ones, ultimately leading to a significant improvement in overall detection performance. By incorporating metadata into the training process, we aim to enhance the network's ability to have better accuracy

on different conditions encountered in maritime surveillance. The metadata will provide valuable contextual information about the image capture conditions, such as altitude, angle, and time of day. This could potentially contribute to the network's robustness and enhance its ability to exhibit invariance across various scenarios.

4.2 Hypothesis and Research Questions

The primary aim of this work is to tackle the difficulties discussed before. We formulated the hypothesis as follows:

There is a way to improve the model's ability to detect objects in images from maritime environments without or with minimal overhead in complexity by resorting to metadata gathered from UAV flight.

This work will be guided by research questions (RQ) to test the above hypothesis. To this end, research questions were created as follows:

RQ1. Can the performance of maritime object detection neural networks be improved by incorporating metadata containing flight information?

There are different ways of incorporating metadata in a network. Some of them can provide better results than others, so finding the best solution to integrate the data in this context is one of the questions we need to answer.

RQ2. Can we incorporate metadata containing flight information without increasing the neural network complexity?

This question aims to investigate if our methodology can also ensure the good performance of the neural network. Creating a way to use metadata without increasing the complexity of our network is essential.

RQ3. Could we apply our methodology to integrate metadata easily on other object detection networks?

This question aims to investigate the potential application and transferability of the developed methodology for integrating metadata into various object detection networks, with the ultimate goal of improving the performance of any arbitrary network.

By verifying the hypothesis and answering the previous research questions, researchers will have new ways of exploring object detection in images from UAVs. As discussed before, object detection in maritime environments is a difficult task, mainly due to the challenging environments in the images and the need to run algorithms in real time on embedded computing platforms.

4.3 Methodology

Zelkowitz et al. [88] defined four experimental categories for validating experimental models, which are listed below.

- **Scientific method** - Scientists develop a theory to explain a phenomenon. They propose a hypothesis and test alternative variations while collecting data to verify the hypothesis.
- **Engineering Method** - Engineers develop and test a solution to a hypothesis. The solution is improved based on test results until it requires no more improvements.
- **Empirical Method** - A statistical method validates a given hypothesis. It may not involve a formal model or theory to describe the hypothesis, and data is collected to confirm the hypothesis.
- **Analytical Method** - A formal theory is developed, and results derived from that theory can be compared with empirical observations.

In our work, we will use both engineering and empirical methods of validation. We first use the engineering method to build an architecture for object detection that integrates metadata. Subsequently, an empirical method will be employed to evaluate the performance of our methodology and validate our hypothesis.

4.4 Summary

As we saw in Section 4.1, one of the gaps in the literature is that the existing algorithms to detect maritime objects don't make use of metadata, even though there are already some datasets that contain this information. Based on this, we formulated the hypothesis and the research question this work aims to answer in Section 4.2. To end this chapter, we describe the methodology to validate the proposed hypothesis in Section 4.3, where we came to the conclusion that we are going to use the engineering and empirical method.

Chapter 5

Exploratory Data Analysis

In this chapter, we perform an exploratory data analysis of the selected dataset. We begin by introducing the SeaDronesSee Dataset in Section 5.1, emphasizing its relevance for our research because it comprises real data and available metadata. Following that, in Section 5.2, we delve into a comprehensive examination of the data, exploring its various characteristics and explaining the processing techniques applied.

5.1 SeaDronesSee Dataset

Given that our approach focuses on the use of metadata to improve detection, it is essential that the selected dataset provides access to that information. As found out during our literature review, both Seadronessee [44] and DGTA-SeaDronesSee [56] had this metadata available, so the decision lies in choosing between these two options. We ultimately chose the SeaDronesSee dataset due to the fact that it provides real images instead of synthetic ones.

SeaDronesSee [44] is a large dataset designed to help develop systems for search and rescue systems using UAVs in maritime scenarios. This object detection dataset consists of 14,227 images, with the majority of them having a size of 3,840×2,160 pixels. The training set is composed of 8,930 images, the validation set of 1,547, and the test set of 3,750. The task is to detect objects of high interest in maritime environments, being the ones present in this dataset swimmers, boats, jet skis, buoys, and lifesaving equipment. Ground-truth bounding boxes and metadata are available in a Javascript Object Notation (JSON) file. The annotation file regarding object detection data can be seen in Appendix A.1, while the annotations for the metadata are provided in Appendix A.2. However, a subset of the images in the dataset is missing metadata. Out of all images, only 6,212 images in the train set have this information, along with 1,093 in validation images and 2,692 in the test set. Furthermore, it is crucial to assess the utility of each variable present in the metadata associated with the images, which will be discussed in the following section.

The creators of the dataset have not made the true labels publicly available. As a result, in order to assess the model’s performance, they have set up an external website¹ where participants can

¹The submission platform can be accessed at seadronessee.cs.uni-tuebingen.de.

submit their predictions for evaluation. The evaluation protocol uses the metrics mAP, mAP@50, mAP@75, AR1, and AR10 from the COCO [31]. The mAP metric measures average precision over a range of IoU thresholds (0.50 to 0.95 with a step size of 0.05), providing an overall accuracy assessment. mAP@50 focuses on accuracy at an IoU threshold of 0.50, while mAP@75 evaluates at an IoU threshold of 0.75. Regarding average recall, AR1 calculates recall when limited to one detection per image, and AR10 measures recall up to ten detections per image. Since the dataset consists of images with a maximum of ten objects, AR10 can be considered a comprehensive recall measurement for all the objects in the dataset.

5.2 Data Analysis and Processing

In order to gain valuable insights from the SeaDronesSee dataset, we analyzed the metadata on the training, validation, and test set. We examined the different variables available within the dataset and discerned the relevance of them. As a result, we identified variables that are pertinent to our approach and others that are deemed irrelevant.

5.2.1 Extraneous Variables

Our analysis began by examining the timestamps of the captured images. To gain a clearer understanding of the time at which the images were captured, we constructed a histogram, as depicted in Figure 5.1. The diagram shows that all the images were taken during a restricted time interval, between 10 A.M. and 3 P.M. It is also important to note that this information is not available on the test set.

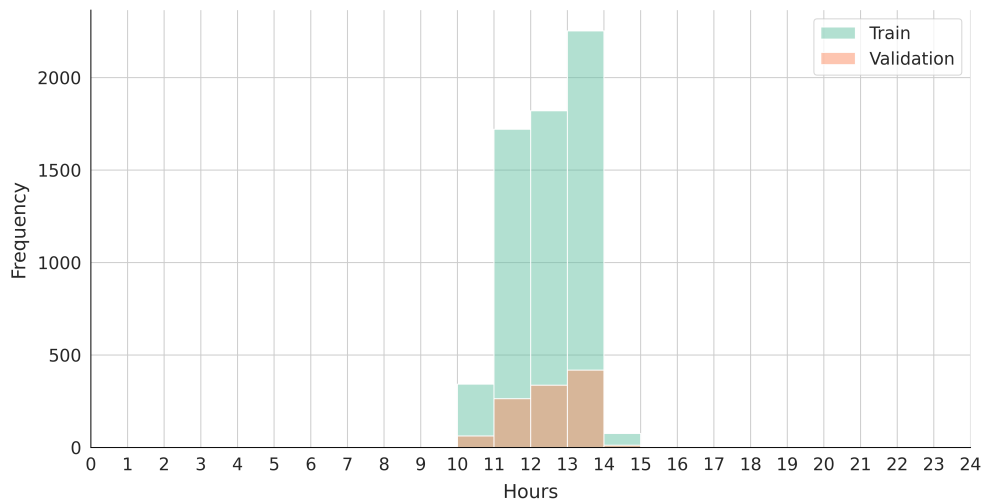


Figure 5.1: Distribution of time of the day in images.

This time frame covers only daylight hours, which imposes certain limitations. We miss out on more challenging scenarios by not having any images of night conditions. Object detection

algorithms trained solely on daytime images may struggle to detect objects in low-light or night conditions, where the environment undergoes significant changes in illumination and object appearance.

The inclusion of time-related features between frames has the potential to improve the performance of object detectors. By exploring the temporal aspect of the captured images, we can gain insights regarding changes and movement of objects occurring over time. This is particularly valuable in scenarios where the environment is subject to variations, like the one we are working on. The speed of the UAV is also another time-related feature that is available in the metadata. With this information, we can have additional context about the motion and trajectory of the captured images and easily capture more information between consecutive frames. However, since this would lead to a more complex network and the data regarding the time at which pictures were taken are not available on the test set, we are not going to use this approach.

In addition, we conducted an analysis of the geographical locations where the images were taken by examining their latitude and longitude coordinates. This revealed that all the images were captured at a lake in Germany. Having the day, time, and location of the images, we thought to investigate the weather conditions by requesting the timestamp and location to a weather **Application Programming Interface (API)**. However, before searching for weather services with a publicly available **API**, due to the limited size of the dataset, we manually inspected the images and observed that they exclusively represented good weather conditions. With this in mind, the approach of getting the weather through an **API** was discarded since it would not provide any relevant result. Nevertheless, exploring this information in datasets that comprise different weather conditions could be valuable for future research. Incorporating this extra variable to be used with our methodology would probably enhance the robustness of the networks to adverse weather conditions.

5.2.2 Relevant Variables

In addition to these previously irrelevant variables, the dataset also has useful metadata, which we can use with our methodology. The ones we found relevant were the camera angle and the altitude at which the images were taken. Understanding more about these variables and their distribution would help to assess their importance, gaining more insights into the conditions present on the dataset since the variation of altitude and angle greatly changes the aspect of the objects present in images.

The altitude variable represents the vertical distance between the aircraft and the sea level at the moment of image capture. The altitude at which the pictures were taken influences the object size, scale, and level of detail, meaning that higher altitudes provide a more challenging time at detection. Figure 5.2 provides valuable insights into the distribution of altitudes at which the images were captured. The histogram visually represents the frequency of images captured at different altitude ranges. Upon careful examination of the histogram, we observe that the majority of images were taken around the altitude of 50 meters. Furthermore, it is worth acknowledging

that there is a relatively lower number of images taken at higher altitudes. This indicates that capturing images at higher altitudes was less frequent or less accessible during the data gathering.

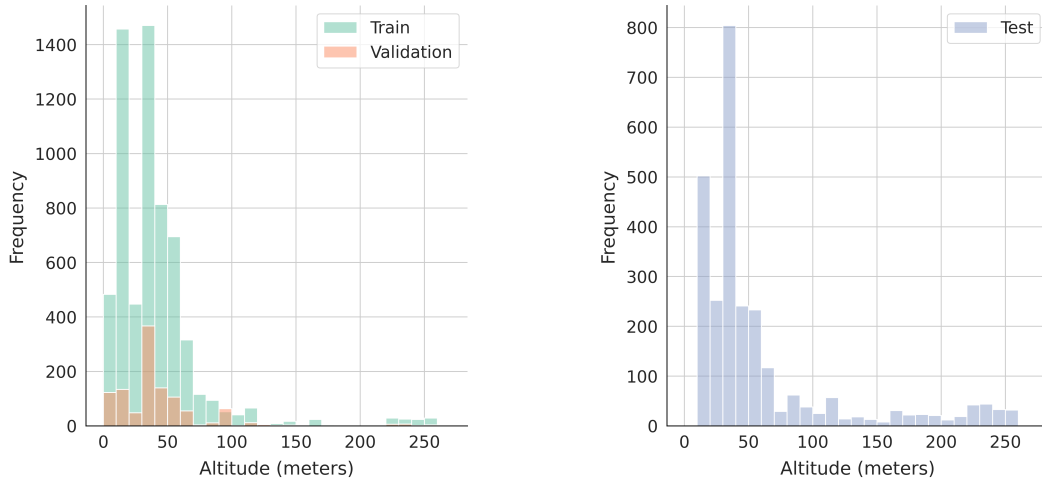


Figure 5.2: Image altitude distribution.

We proceeded to divide the altitude into three intervals, low, medium, and high, where each had the same number of images. Each interval corresponds to a distinct altitude range and will be used by our network in order to become more robust. Figure 5.3 provides visual examples of an image on each altitude scale.

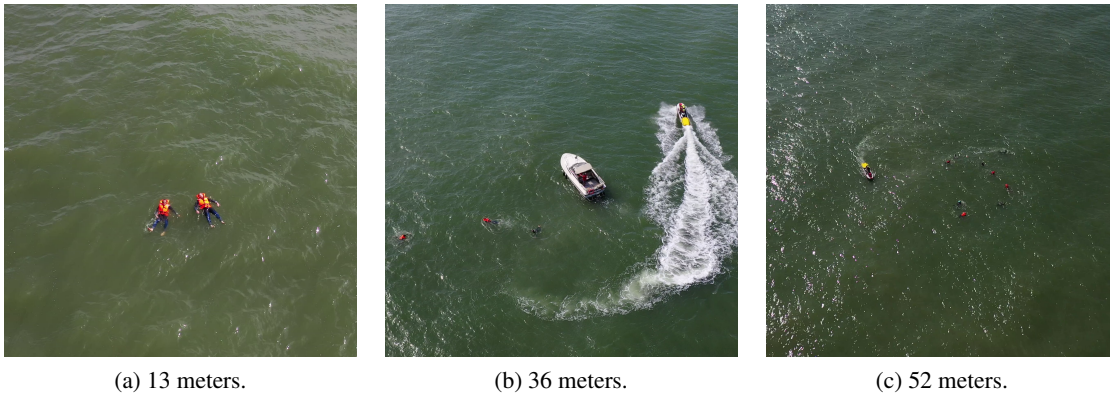


Figure 5.3: Images captured from UAVs showcasing diverse altitude ranges.

On the other hand, the angle variable refers to the orientation from which the images were captured. Angle influences the perspective of the objects, affecting the features that can be extracted from the images. Figure 5.4 provides a visualization of the distribution of angles at which the images were captured, allowing us to observe the frequency of images within different angle ranges and providing insights into the predominant angle variations present in our dataset. When taking into account all the images in the dataset, most of them are taken from a horizontal and

slightly tilted perspective. In contrast, there is a noticeably lower number of images with a top-down perspective, where the angle exceeds 65 degrees, which provides a bird's-eye view. Upon careful analysis of the histograms, we can clearly see that most images in the train set contain a significant proportion of images with angles below 40 degrees. In contrast, the test set has a higher number of images with 90 degrees.

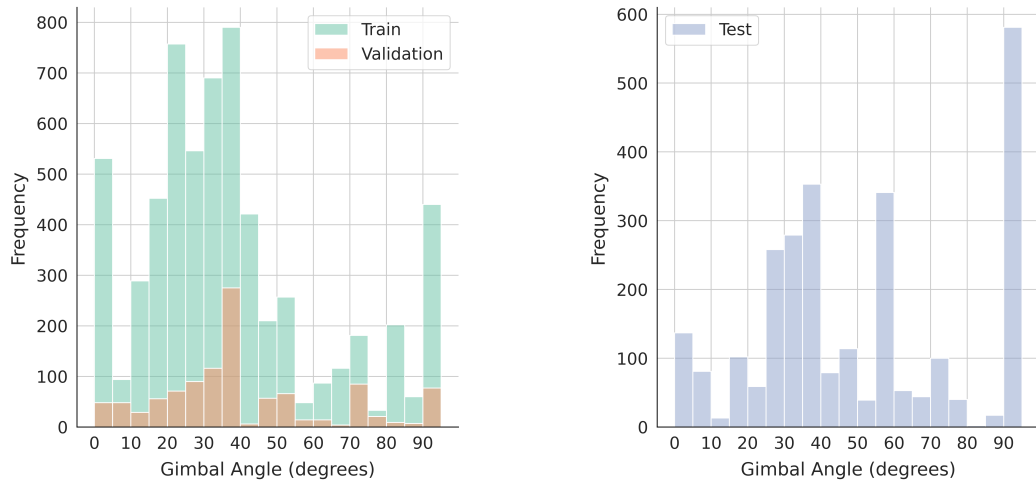


Figure 5.4: Image angle distribution.

Similarly to the altitude, we divided the angle variable into three groups based on the respective camera angles. However, we did not divide the angles into equidistant intervals, where each interval would have the same frequency of occurrence. Instead, the first group encompassed angles between 0 and 35 degrees, the second group covered angles from 35 to 65 degrees, and the third included angles greater than 65 degrees in the train set. Figure 5.5 showcases samples from the dataset, illustrating the angle variation.

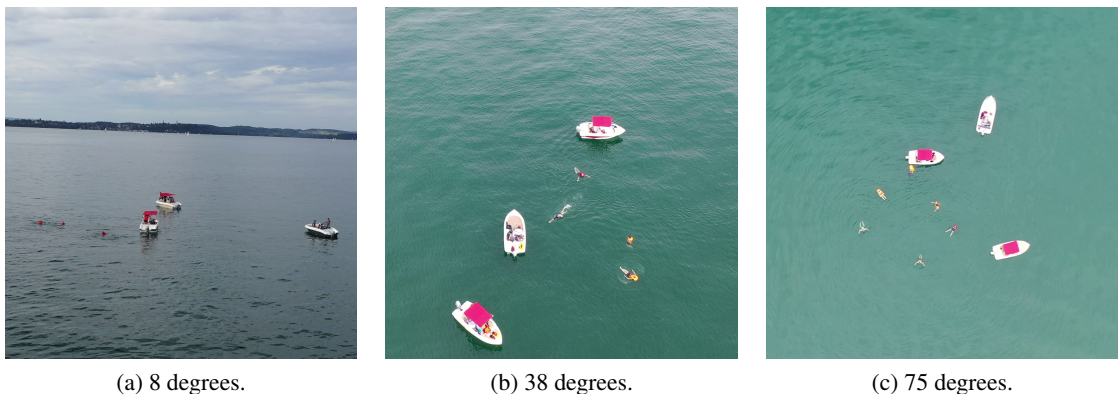


Figure 5.5: Images captured from UAVs showcasing diverse angle ranges.

Both altitude and angle are important in determining the orientation and viewpoint of the objects within the images, providing crucial context, and will significantly contribute to improving

the robustness of our network for object detection. By leveraging this metadata, we can enhance the model's performance and enable it to handle variations in these factors more effectively, thus mitigating potential biases and challenges.

We noticed a pattern within the dataset during our analysis of the altitude and angle variables. We found out that the images captured at higher altitudes tend to have the camera angled downwards, while images taken at lower altitudes exhibited a tendency for the camera to be horizontal. To examine the potential relationship between angle and altitude variables, we conducted an analysis of their distribution. A scatter plot, depicted in Figure 5.6, was used to visualize this relationship, revealing that the altitude and camera angle are not orthogonal. This pattern suggests a logical relationship between the drone's altitude and the camera orientation. Higher-altitude images, where the camera is angled downwards, likely capture a broader view of the surrounding area from an elevated position. On the other hand, lower-altitude images with the camera pointed toward the front help to capture more objects present in the sea.

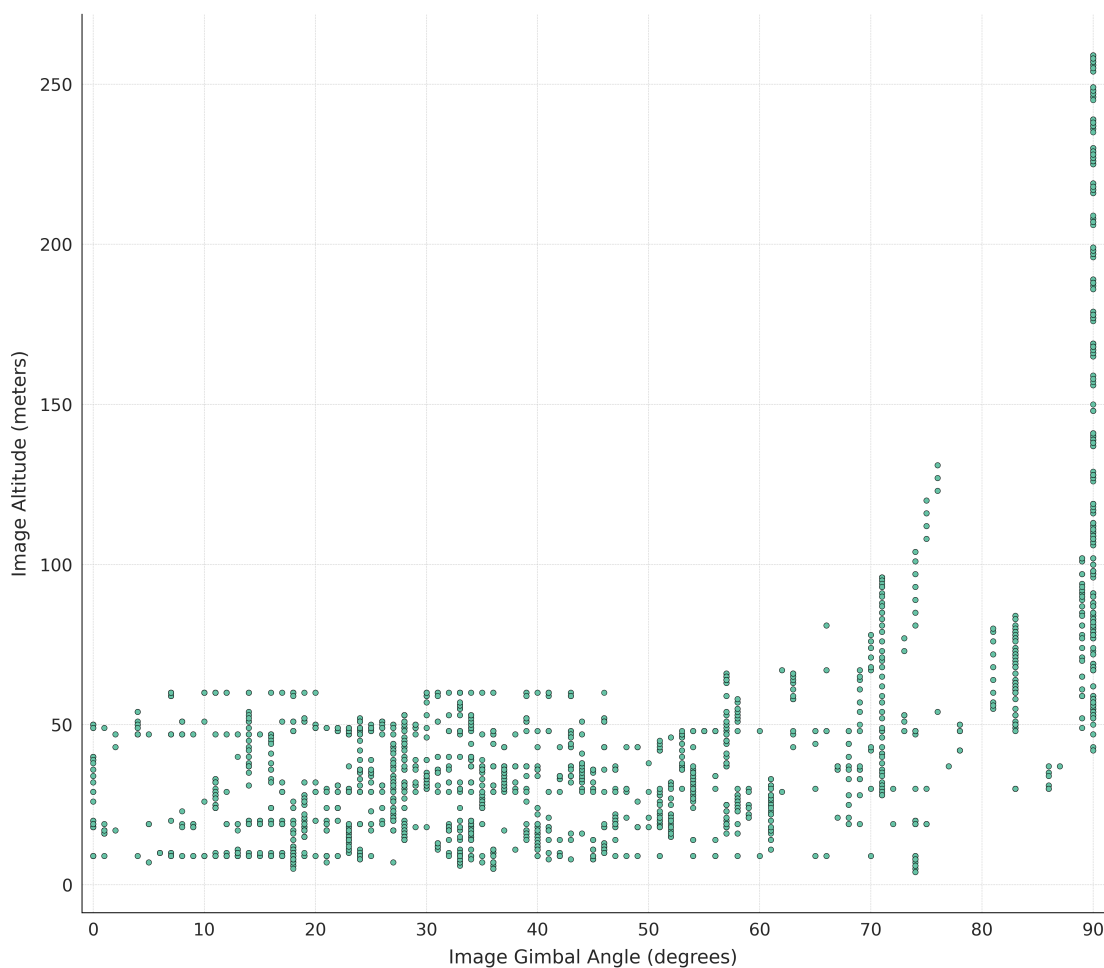


Figure 5.6: Scatterplot between altitude and angle.

5.3 Summary

In this chapter, we delved into the data that we are going to use in this work. In Section 5.1, we chose the SeaDronesSee dataset as the primary dataset for our analysis due to its comprehensive metadata coverage and for using real data. After that, in Section 5.2, we analyzed the availability and relevance of the available metadata present in our dataset, hence its importance. We found out that some of the information available in the metadata is not relevant, and the dataset is not very representative of diverse maritime situations, where only good weather conditions during daylight hours are present. Adverse weather and sea conditions, such as fog or rain, can significantly impact maritime operations and introduce additional obstacles for object detection models. These conditions were unavailable on this dataset and often result in degraded visibility and changed object appearance, making accurate detections more challenging. After conducting our analysis, we concluded that the altitude and angle variables hold significant importance. In the forthcoming chapters, we will integrate these variables into neural networks to explore whether leveraging metadata can improve their performance.

Chapter 6

Architecture

The chapter begins with an introduction to the selected baseline and the proposed architecture in Section 6.1. The implementation details of our architecture are described in Section 6.2, and in Section 6.3, we outline the training procedure and discuss the associated loss functions. This chapter provides a comprehensive understanding of the architecture, its implementation, and the training methodology of our methodology to integrate metadata in neural networks.

6.1 Model Overview

This section provides an overview of the model, introducing the architecture we are going to use during this work. In Section 6.1.1, we start by presenting our baseline model, which serves as a reference point for comparison. Following that, Section 6.1.2 delves into the proposed architecture. We discuss the underlying principles and motivations that guided its design, highlighting its unique features and advantages.

These two sections lay the foundation for a comprehensive understanding of the model's development and the subsequent subsections that provide more detailed insights into its components and training procedures.

6.1.1 Baseline

As mentioned earlier during the data analysis, we have selected the SeaDronesSee dataset for our research. The authors of this dataset conducted a workshop [89], where they chose YOLOv7, which has 36.9M parameters, as the baseline. YOLOv7 is a state-of-the-art algorithm that has demonstrated outstanding performance in object detection tasks and is known for its real-time detection capabilities. Therefore, it aligns well with our research objectives since we need efficient algorithms to be deployed on UAVs, and we aim to establish a solid foundation for our subsequent experiments and evaluations. The baseline model will serve as a reliable benchmark to evaluate and compare the performance of our proposed architecture.

6.1.2 Proposed Architecture

In this study, our objective is to investigate the potential impact of variations, specifically the altitude and angle we had explored before, on the performance of object detection algorithms in maritime environments. We aim to extract valuable information regarding these characteristics by thoroughly examining the available metadata. Our goal is to have a model that preserves object-related information while discarding or de-emphasizing the influence of these variations unrelated to the objects. By doing so, we ensure that the model consistently represents objects based on their inherent properties, improving the accuracy of the object detection task. Our architectural design was built upon the foundations established by Ganin et al. [35], who introduced an adversarial learning algorithm for domain adaptation. Furthermore, we drew inspiration from the work of Zhenyu Wu et al. [40], who applied a similar approach in the context of UAV images within a terrestrial environment. The high-level diagram of the architecture we employed can be seen in Figure 6.1.

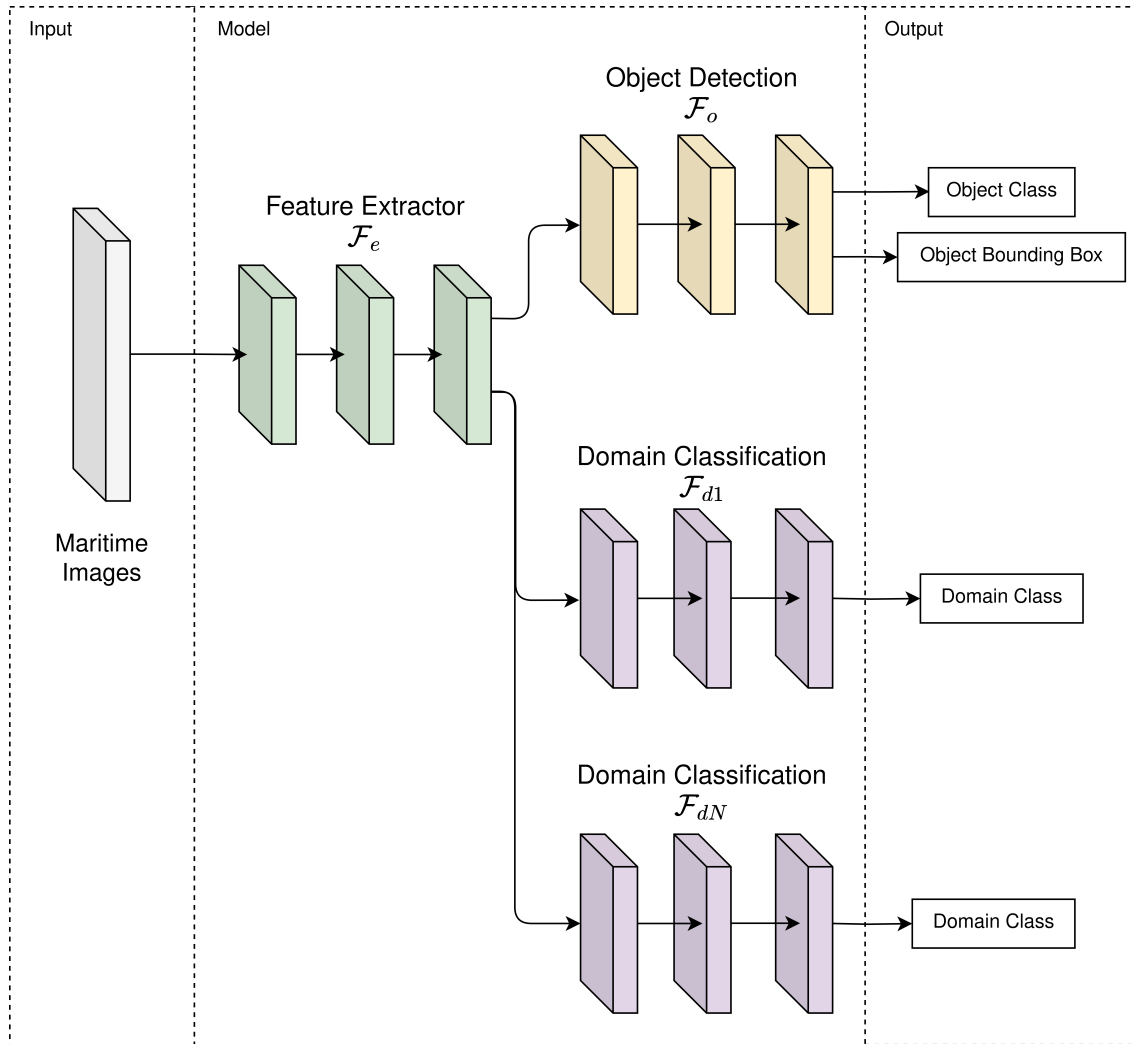


Figure 6.1: High-level diagram of the proposed architecture.

The goal of this network is to develop a domain-invariant model where the representations extracted should discard a significant amount of domain-specific information. Consequently, it is crucial to note that the domain classification network cannot accurately classify the original domain that the image contains, leading to a problem in which:

- A domain classification network F_d receives latent representations from the feature extractor F_e and will try to predict the domain associated with a given image.
- An object detection network F_o receives latent representations from the feature extractor F_e and will try to predict the objects associated with a given image.
- The feature extractor F_e will attempt to deceive the domain classification network F_d while providing high-quality representations to ensure optimal performance of the object detection network F_o .

Our approach aims to maximize the object detection performance while simultaneously making the domain classification incapable of accurately detecting the domain. Therefore, the main objective of the feature extractor is to reduce the dissimilarity between similar images captured in different domains. This is achieved by aligning the distributions of encodings across these domains. Additionally, our architecture is designed to accommodate an arbitrary number of domain classification networks since it is important to consider the possibility of multiple domains.

Acknowledging that this deliberate emphasis on making the model invariant to domain characteristics may have potential drawbacks is essential. The model may inadvertently overlook being invariant to these variables and struggle to detect objects, which is the main task. To mitigate the potential impact of these inherent characteristics on the object detection task and emphasize object-related information, we will not use domain classification networks every epoch, using it after a predefined number of epochs of training without this module, which will be defined through experimental testing.

Considering that our model needs to be efficient at inference time, we only use the metadata for training. During inference, the feature extractor and object detection network are used to perform predictions while the domain classification network is discarded. This makes the network more lightweight, which is necessary for deploying on embedded platforms present in UAVs. To test and evaluate the network's performance, we have employed two distinct techniques described below.

Adversarial Model Figure 6.2 shows an example of inference using the trained model end-to-end. This model has the flexibility to be trained on either a single domain or multiple domains, thanks to the effective utilization of the adversarial methodology during the training process.

Ensemble Model Figure 6.3 a way of doing inference with a model ensemble. Each model is first trained in just one domain. The ensemble model then aggregates the prediction of each base model and results in one final prediction for the unseen data. We will employ this method because when using multiple domain classification networks, the gradients may be canceling each other.

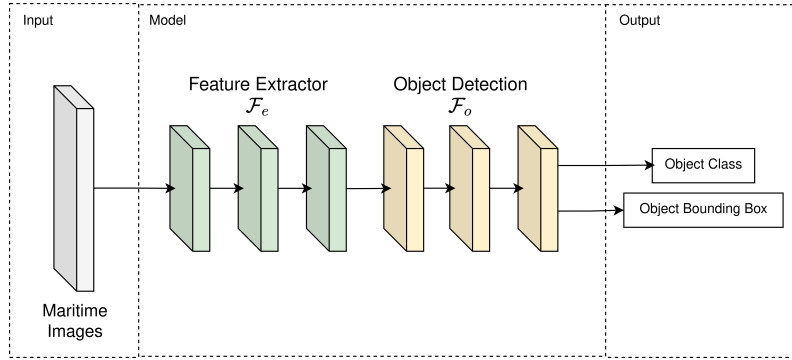


Figure 6.2: Inference method.

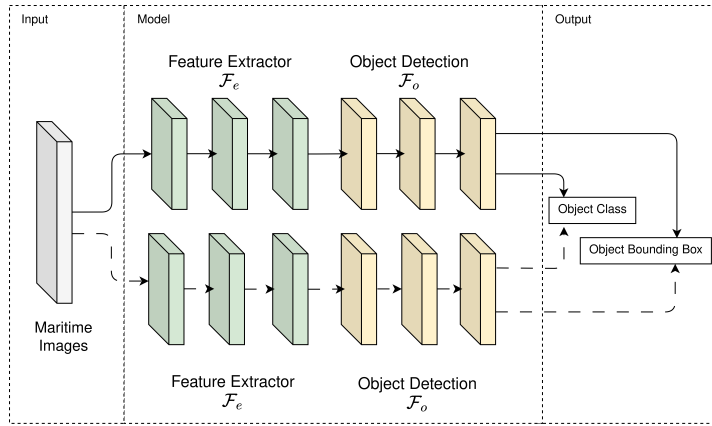


Figure 6.3: Inference method using ensemble.

6.2 Implementation Details

As mentioned before, we will adopt YOLOv7 as the foundation for our architecture. In our implementation, F_e and F_o are built upon YOLOv7, where F_e comprises the first 40 layers of the backbone, and F_o is built with the remaining 10 layers of the backbone along with the head. Furthermore, we introduced a model responsible for domain classification, F_d , which consists of convolutional and fully connected layers. For each domain, we create an additional F_d module.

One problem that arises with the use of YOLOv7 is the high number of image augmentations that are responsible for making this network more robust and high performing, such as mosaic augmentation, which combines four different images into one. As a result of this blending, it becomes impossible to discern which domain the image belongs to, and F_d will be unable to classify the corresponding domain.

To address this challenge, we adopted a two-pass approach during the feature extraction stage, splitting the training process. The feature extractor F_e is fed both with the augmented images and vanilla images. As depicted in Figure 6.4, the augmented images are used to perform object detection, while the ones without augmentation are going to be used to classify the domain.

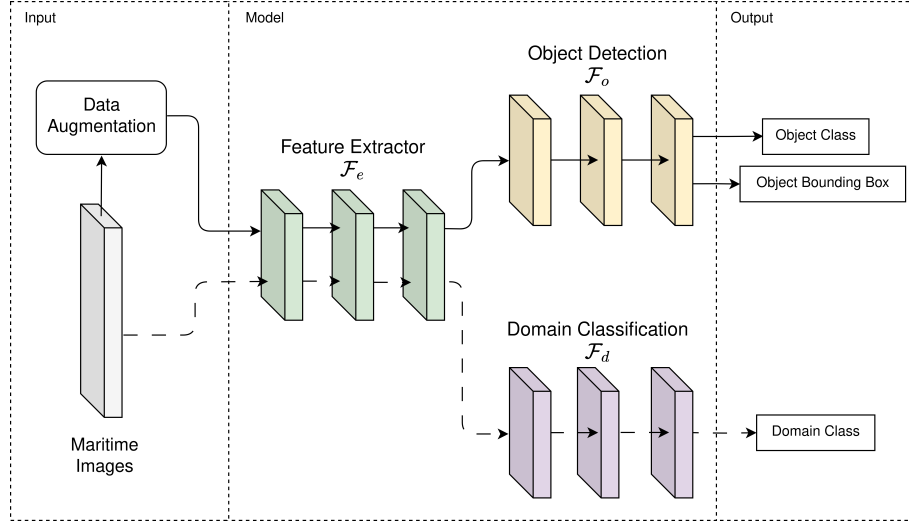


Figure 6.4: Diagram of our architecture with image processing.

6.3 Training and Losses

The training loop of our methodology comprises three distinct stages, each contributing to the overall process of training our model. This iterative training process aims to improve the model's ability to accurately detect objects in maritime environments while disregarding domain-specific characteristics.

6.3.1 First Stage

During the first stage of our methodology, we focus on training the domain classifier to distinguish between different domains. As depicted in Figure 6.5, the object detection network is not used, and the feature extractor is frozen. Freezing a module means its parameters are not updated during the training. Since the adversarial network needs to be strong enough, we train the domain classification until we meet a minimum of 70% validation accuracy or 90% training accuracy or after a maximum number of epochs to avoid overfitting. In this stage, we exclusively work with non-augmented images from the dataset. The input data is processed through the feature extractor, and the resulting output is subsequently subjected to image domain classification, outputting the predicted probability p for each domain. The loss that the domain classification tries to minimize is shown in equation (6.1). Here, gt_i represents the true probability of class i (one-hot encoded), and p_i represents the predicted probability of class i . Following this process, we obtain a robust domain classifier network capable of accurately predicting the domain labels.

$$L_{domain}(gt, p) = - \sum_{i=1}^N gt_i \log(p_i) \quad (6.1)$$

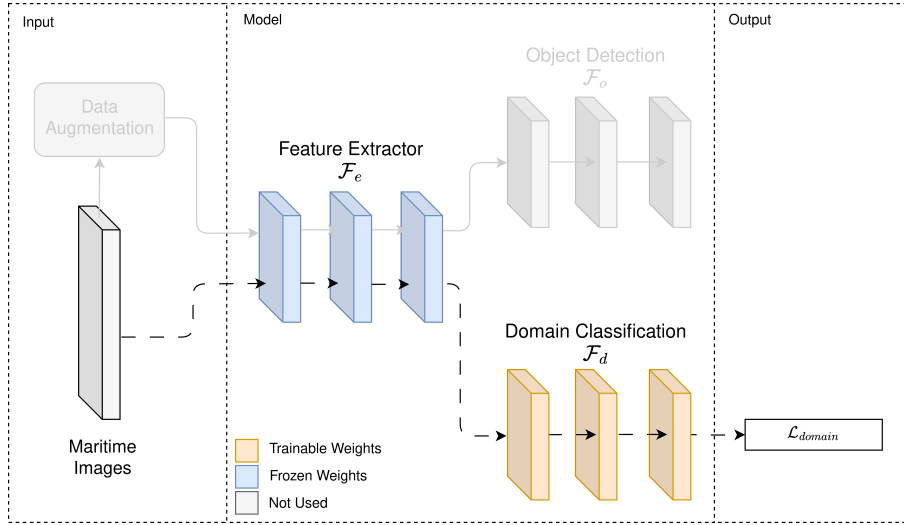


Figure 6.5: First stage of our training loop.

6.3.2 Second Stage

During the second stage, the domain classifier network is frozen. The feature extractor is updated by the adversarial and object detection loss, while the object detection network is updated just by the object detection loss. The input data is processed again through the feature extractor, and the resulting output is passed through two subsequent branches. Augmented images are fed through the feature extractor and then to the object module, while non-augmented images are fed through the feature extractor and then to the domain classifier, similar to the first stage of the training. This can be seen in Figure 6.6.

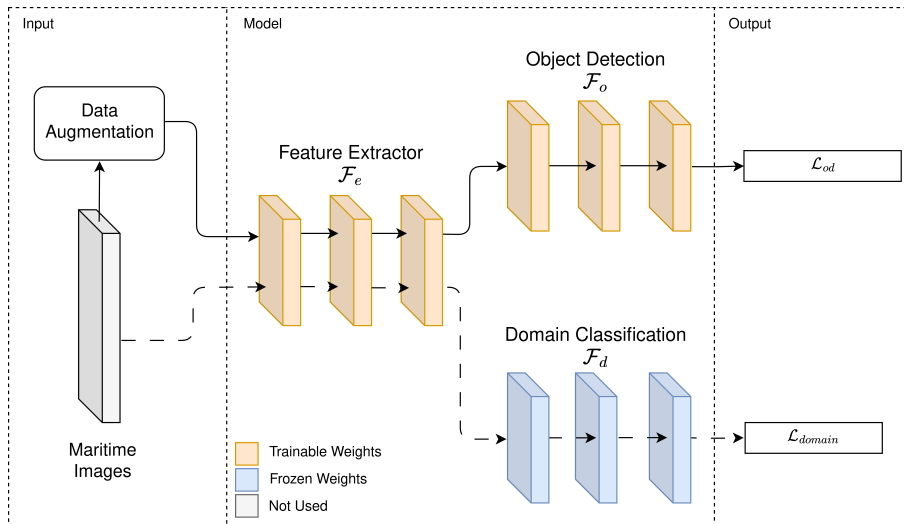


Figure 6.6: Second stage of our training loop.

This helps the encoder to learn features in a way that the predictions of the object detection are as accurate as possible, and the predictions of the domain classification are kept close to a uniform distribution, meaning that this domain classifier is not better than random guessing the

corresponding domain. For this, we use the **Kullback-Leibler (KL)** divergence as the adversarial domain loss. The **KL** divergence serves as a measure of the dissimilarity between the predicted probabilities and the target distribution. It quantifies the difference between the two distributions, with higher values indicating more significant dissimilarity. The **KL** loss is depicted in equation (6.2), being P_i the probability of domain i according to the predicted distribution P , and Q_i represents the probability of domain i according to the uniform target distribution Q . By minimizing the KL divergence, we encourage the predicted probabilities to approach the uniform distribution.

$$\mathcal{L}_{adv}(P \parallel Q) = \sum_i^N P_i \log \left(\frac{P_i}{Q_i} \right) \quad (6.2)$$

The loss regarding object detection is the one used in the YOLOv7, and we denote this loss as $\mathcal{L}_{od}$, which is composed of the bounding box, objectness, and classification loss. The loss function is presented in equation (6.3), which is the sum of the object detection loss $\mathcal{L}_{od}$ and the adversarial loss $\mathcal{L}_{adv}$, which is multiplied by a weighting factor λ . By jointly training these branches, the model will learn domain invariant features.

$$\mathcal{L} = \mathcal{L}_{od} + \lambda \mathcal{L}_{adv} \quad (6.3)$$

6.3.3 Third Stage

During the third stage, the focus is solely on updating the weights of the model based on the object detection task, as can be seen in Figure 6.7. This means that during this stage, the training process is exclusively driven by the object detection loss without considering the adversarial domain loss or any domain-related considerations.

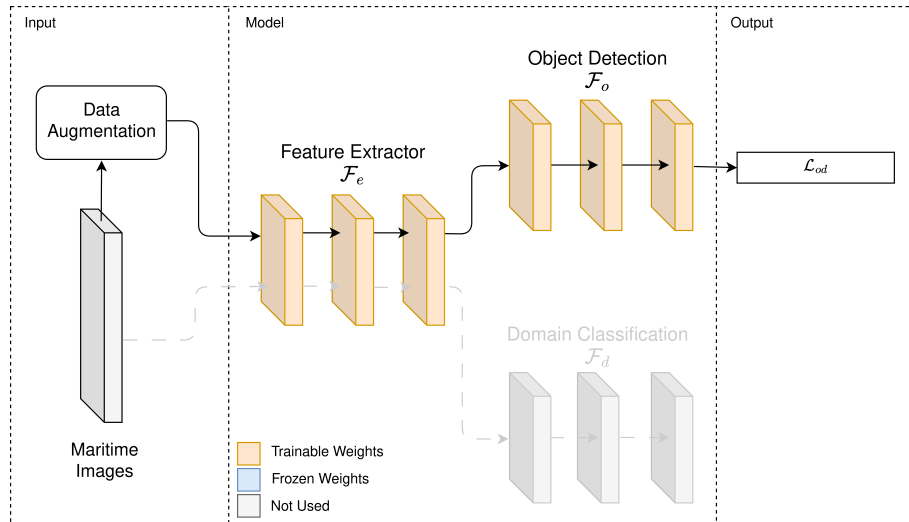


Figure 6.7: Third stage of our training loop.

The decision to use only object detection in this stage is to ensure that the feature extractor learns to extract relevant features related to the images and objects in the dataset. This stage occurs

for a certain number of epochs. Once this stage is completed, it goes back to the first one, and the weights of the domain classifier, optimizer, and scheduler are reset. This reset is necessary because the representations learned by the feature extractor may change over time. By resetting the domain classification module, we avoid potential bias or interference from previous domain-specific information, allowing the module to relearn and adapt to the current domain variations.

6.4 Summary

In Section 6.1, we started by choosing the baseline, which will be the YOLOv7 allowing us to establish a reference performance level to evaluate our architecture. We also presented the designed architecture for our task. We employed an adversarial learning technique to enhance the model's ability to learn invariant representations to be able to perform a more robust detection across different domains. Following the introduction of the architecture, in Section 6.2 we addressed the implementation details clarifying that our approach is built upon YOLOv7. Concluding this chapter, we present comprehensive information regarding the training procedure in Section 6.3.

Chapter 7

Results and Discussion

This chapter presents the main findings and experiments conducted during our research. First, we focus on the reproducibility of our research results in Section 7.1, and perform hyperparameter tuning to establish a baseline in Section 7.2. We then proceed to examine the performance of our architecture in Section 7.3. Furthermore, we investigate the impact of mosaic augmentation on the performance of our network in Section 7.4. Finally, we discuss our hypothesis and the research questions in Section 7.5.

7.1 Reproducibility

In pursuit of the reproducibility of the results, we start by establishing a consistent starting point for all experiments, selecting predefined seeds, and ensuring that the initial conditions for the model training were identical across different runs. To further enhance reproducibility, we followed the PyTorch guidelines to ensure determinism in CUDA operations. By enabling CUDA deterministic settings, we mitigated potential sources of non-deterministic behavior. Reproducibility is essential for scientific improvement because it allows the research community to validate and improve current work. The detailed runs and results of our experiments are available on Wandb, which can be accessed from the GitHub repository ¹ where the code is also stored.

7.2 Baseline and Hyperparameter Tuning

We start our work improving the baseline provided on SeadronesSee Workshop [89] to ensure a solid enough validation for our proposed methodology. The hyperparameters were evolved using evolutionary optimization techniques, which are effective in improving the performance of our model [90]. We evolved them for 350 iterations on top of the default hyperparameters. After each iteration, the runs are ranked by their relative fitness, which is the value we seek to maximize. We used the default fitness metrics from YOLOv7, in which mAP@50 contributes 10% of the weight and mAP contributes the remaining 90%, with Precision and Recall absent. The mutation

¹The code is available at github.com/ipleiria-robotics/maritime-metadata-adaptation

has a probability of 80% and a 0.04 variance. The evolved hyperparameters obtained from this evolutionary optimization can be found in Appendix B. Each iteration consisted of 20 epochs, allowing the model to converge to optimal results without overfitting. The image size used was 640, although bigger sizes may have yielded better results. Unfortunately, hardware limitations prevented us from exploring larger image sizes. To ensure that our results are reliable and not dependent on seed randomness, we run the models on different seeds to ensure robustness. Considering the average and standard deviation of these runs helped us to obtain a more comprehensive understanding of the model performance.

Table 7.1 presents the performance with the default hyperparameters. The average mAP of the five runs is 42.20%, with a standard deviation of 0.24, which indicates the level of dispersion of the performance scores, providing insights into the consistency and reliability of the model. Due to the presence of small objects, the lower mAP@75 score suggests that our model struggles to accurately detect and localize these small objects, possibly due to the limited resolution and detail available. However, the higher mAP@50 score performs better when a less strict intersection over the union threshold is applied.

Table 7.1: Performance using default hyperparameters.

Model	mAP	mAP@50	mAP@75	AR1	AR10
Seed 0	42.20	72.25	43.6	38.66	50.99
Seed 42	41.92	71.00	42.82	37.77	50.08
Seed 821	42.16	73.07	43.73	38.32	50.91
Seed 1765	42.50	71.47	44.06	38.43	51.55
Average	42.20	71.95	43.55	38.30	50.88
Standard Deviation	0.24	0.91	0.53	0.38	0.61

Table 7.2 displays the model performance using evolved hyperparameters. The average mAP for these five runs stands at 42.57%, with a standard deviation of 0.34. It is 0.37 better than the baseline with default hyperparameters. Since the model has a higher mAP, but there is a slight decrease in the recall values, this implies that the model is more confident in its predictions. Even though the results weren't significantly better, they exhibited faster convergence, as depicted in Figure 7.2. From now on, our baseline will be the one using evolved hyperparameters.

Table 7.2: Performance using evolved hyperparameters.

Model	mAP	mAP@50	mAP@75	AR1	AR10
Seed 0	43.02	71.86	45.00	38.26	50.17
Seed 42	42.55	71.42	44.29	37.54	49.61
Seed 821	42.18	70.81	44.46	37.08	49.25
Seed 1765	42.53	71.25	44.94	37.52	49.41
Average	42.57	71.34	44.67	37.6	49.61
Standard Deviation	0.34	0.43	0.35	0.49	0.40

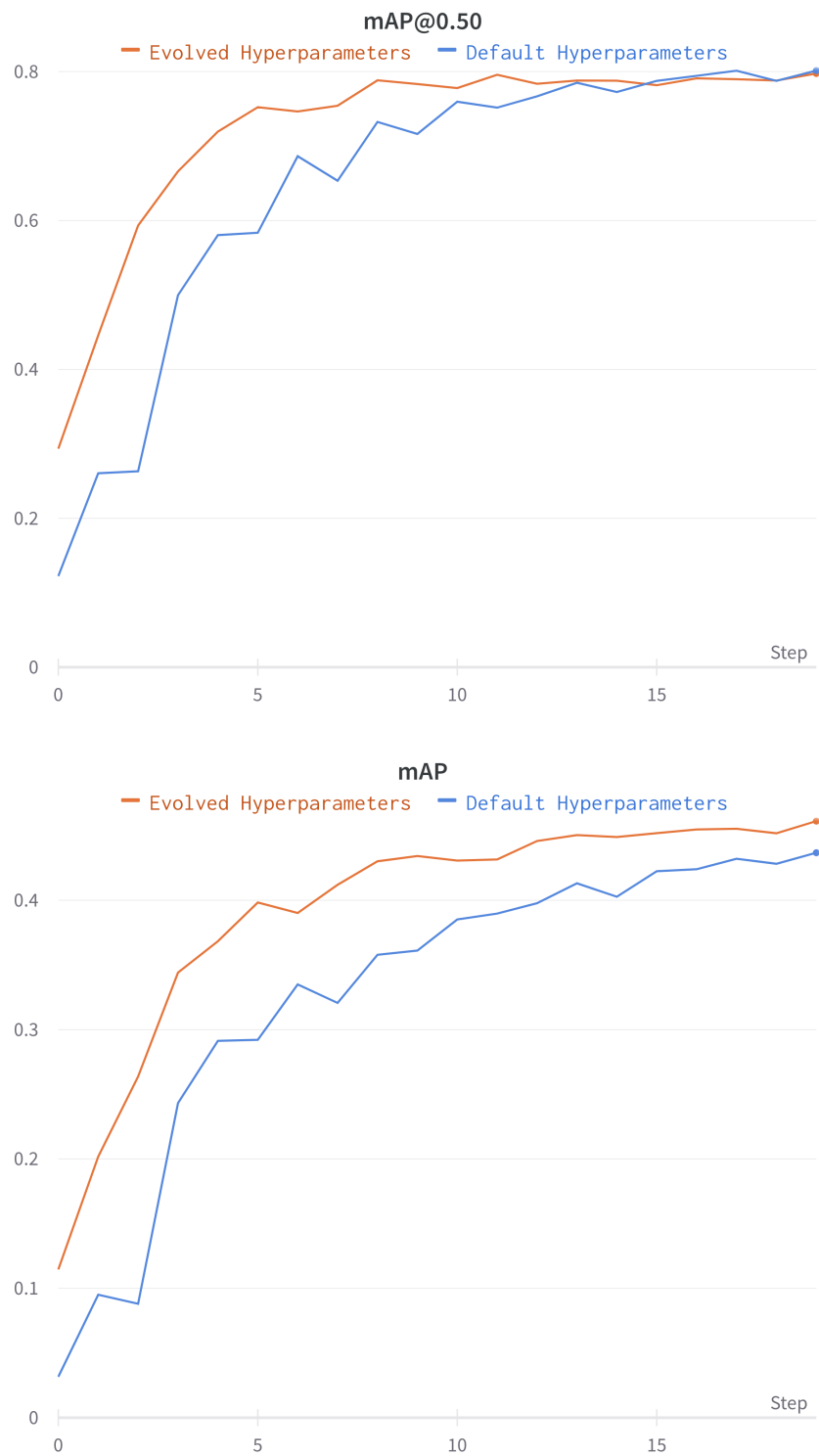


Figure 7.1: Evolution of mAP and mAP@50 over the epochs.

7.3 Domain Adversarial Learning

Once we had established a baseline model, our next step was to evaluate the performance of our proposed architecture. In Section 7.3.1, we conducted an evaluation of the model performance with respect to altitude and angle variables using just a single domain. In Section 7.3.2, we analyzed the model performance when considering both domains. Finally, we assessed the performance of the model ensemble in Section 7.3.3.

7.3.1 Single Domain

To start our experiments, we carefully selected values for N , the number of epochs that the model will train before using the domain adversarial network again, and for λ , the weighting factor for our adversarial loss.

As mentioned during the data analysis, we categorized the altitude variable into three distinct domains: low, medium, and high. This division enables our model to learn and distinguish between the different features that exist at these altitude intervals. We performed some experiments using different combinations of N and λ , as can be seen in Table 7.3. We started by experimenting with $N = 1$, but the obtained results were unsatisfactory and subsequently disregarded, leading us to choose to test the values 4 and 8 with a different λ . Although other values could have been tested, we opted to stick with these two values due to the large number of tests that needed to be conducted.

Table 7.3: Performance using altitude domain.

N	λ	mAP	mAP@50	mAP@75	AR1	AR10
8	0.001	43.85	74.28	45.85	38.39	51.57
	0.005	42.88	71.75	45.30	37.45	49.67
	0.01	42.75	72.24	44.20	37.61	49.93
4	0.001	43.17	73.47	45.07	37.89	50.50
	0.005	43.09	71.60	46.16	37.77	50.23
	0.01	42.52	71.95	44.48	37.63	49.58

Upon analyzing the results, it becomes evident that using a $\lambda = 0.001$ and employing the adversarial domain approach every 8 epochs yield the best results, with an mAP of 43.85%. We selected this model and conducted additional runs using different seeds to obtain the average and standard deviation, as shown in Table 7.4, following a similar methodology as the baseline to ensure that our results are reliable. We obtained an average mAP of 43.05%, which signifies a marginal increase of 0.48 over the baseline average. However, when looking at mAP@50, the increase in our methodology is 1.41. At mAP@75 the minimal gain can happen because objects with small dimensions can be easily missed or misclassified, leading to lower performance. This reduced accuracy at this threshold could also be caused due to small image resolution present in our images.

Table 7.4: Performance using the best parameters regarding altitude.

Model	mAP	mAP@50	mAP@75	AR1	AR10
Seed 0	43.85	74.28	45.85	38.39	51.57
Seed 42	42.17	71.74	43.55	37.26	49.65
Seed 821	42.96	72.26	44.75	37.48	49.88
Seed 1765	43.23	72.70	46.00	37.70	50.25
Average	43.05	72.75	45.04	37.71	50.34
Standard Deviation	0.70	1.10	1.14	0.49	0.86
Baseline Average	42.57	71.34	44.67	37.60	49.61
Improvement over Baseline Average	+0.48	+1.41	+0.37	+0.11	+0.73

Once we have successfully trained a model to exhibit invariance towards different altitudes, we proceed to investigate the influence of the angle variable. Similar to the altitude, the angle is partitioned into three distinct intervals, each representing a distinct viewpoint. The results obtained from this testing different values of N and λ are depicted in Table 7.5. The highest achieved performance was 43.77% when using $N = 8$ and $\lambda = 0.005$.

Table 7.5: Performance using angle domain.

N	λ	mAP	mAP@50	mAP@75	AR1	AR10
8	0.001	42.96	71.83	45.54	37.76	50.29
	0.005	43.77	74.15	45.56	38.84	51.23
	0.01	43.02	72.05	45.45	37.63	50.07
4	0.001	43.19	72.72	45.07	37.92	50.09
	0.005	43.36	72.39	45.65	38.27	50.5
	0.01	40.91	69.37	41.82	36.10	48.34

We run this combination across different seeds and got an average mAP of 43.24% with an increment of 0.67 over the baseline, as can be seen in Table 7.6. Similar to the altitude results, we observed higher gains at mAP@50, where we had an improvement of 1.30. The increase in performance at mAP@75 is relatively small, with an improvement of only 0.47.

Table 7.6: Performance using the best parameters regarding angle.

Model	mAP	mAP@50	mAP@75	AR1	AR10
Seed 0	43.77	74.15	45.56	38.84	51.23
Seed 42	43.51	73.45	45.34	37.91	50.45
Seed 821	42.41	71.13	44.08	37.62	50.01
Seed 1765	43.28	71.83	45.56	38.18	50.47
Average	43.24	72.64	45.14	38.14	50.54
Standard Deviation	0.59	1.40	0.71	0.52	0.51
Baseline Average	42.57	71.34	44.67	37.60	49.61
Improvement over Baseline Average	+0.67	+1.30	+0.47	+0.54	+0.93

7.3.2 Multiple Domains using Adversarial Learning

Testing with both domains simultaneously is expected to result in improved performance since the network will exhibit invariance to both domains simultaneously, rather than focusing on just one domain. For this approach, two domain classification modules were used one for the altitude and one for the angle, and a different λ was employed for each domain. We used $N = 8$ since this value provided the best results on each domain individually. We then tested with different values of $\lambda_{Altitude}$ and λ_{Angle} , as can be seen in Table 7.7.

Table 7.7: Performance using multiple domains.

N	$\lambda_{Altitude}$	λ_{Angle}	mAP	mAP@50	mAP@75	AR1	AR10
8	0.001	0.001	42.35	71.31	44.53	37.17	49.57
	0.001	0.005	42.33	70.45	44.64	37.08	49.44
	0.005	0.001	43.37	72.72	45.01	37.93	50.49
	0.005	0.005	43.61	73.18	45.58	38.15	50.64
	0.0005	0.0005	42.02	71.50	43.42	36.97	49.37

The best mAP obtained was 43.61% using $\lambda_{Altitude} = 0.005$ and $\lambda_{Angle} = 0.005$. However, the obtained mAP in this scenario is lower than the best mAP obtained when considering only a single domain for angle and altitude. We conducted model runs on various seeds using the aforementioned values to ensure a comprehensive comparison with previous experiments and accurately assessed that this result is indeed inferior. The results of these runs are presented in Table 7.8.

Table 7.8: Performance using the best parameters regarding multiple domains.

Model	mAP	mAP@50	mAP@75	AR1	AR10
Seed 0	43.61	73.18	45.58	38.15	50.64
Seed 42	42.83	72.24	44.52	37.64	50.07
Seed 821	42.57	71.13	44.77	36.87	49.38
Seed 1765	42.49	71.40	45.03	37.33	49.44
Average	42.88	71.99	44.98	37.50	49.89
Standard Deviation	0.51	0.93	0.46	0.54	0.60
Baseline Average	42.57	71.34	44.67	37.60	49.61
Improvement over Baseline Average	+0.31	+0.65	+0.31	-0.1	+0.28

Upon analyzing the results, we confirmed that the performance was lower when employing multiple-domain adversarial training than using a single-domain adversarial approach. One possible factor for this occurrence could be the mutual exclusion of gradients since the domains considered (altitude and angle) are not orthogonal with respect to their impact on captured images, as we saw during our data analysis.

7.3.3 Multiple Domains using Model Ensemble

The ensemble model employed in this study also tries to be invariant to multiple domains at the same time. We combine the weights from different pre-trained models using the adversarial learning technique. We integrated the weights obtained from the models that were solely trained on a single domain, either on altitude or angle. To illustrate this with an example, we used the weights from the best angle model of seed 0 and the weights from the best altitude model of seed 0 to form an ensemble model. We repeated this process for the different seeds, and the results of this approach are presented in Table 7.9.

Table 7.9: Performance using ensemble model.

Model	mAP	mAP@50	mAP@75	AR1	AR10
Seed 0	45.12	76.57	46.78	39.93	53.47
Seed 42	43.62	73.87	45.22	38.54	51.44
Seed 821	43.41	73.53	44.76	38.16	51.06
Seed 1765	43.90	74.44	46.10	38.64	51.43
Average	44.01	74.60	45.72	38.82	51.85
Standard Deviation	0.77	1.36	0.90	0.77	1.09
Baseline Average	42.57	71.34	44.67	37.60	49.61
Improvement over Baseline Average	+1.44	+3.26	+1.05	+1.22	+2.24

This ensemble model yielded the best results regarding all techniques. By combining the strengths of multiple models, we enhanced overall performance and achieved moderate gains with a 1.44 improvement on mAP and an improvement of 3.26 on mAP@50 over the baseline model. However, it is important to note that this method increases the computational load, making the model more resource-intensive and less suitable for deployment on embedded devices.

7.4 Influence of Mosaic Augmentation

Even though the results obtained using our architecture were higher than the baseline, they only represented marginal gains. Upon closer examination, we suspected that the use of mosaic augmentation might be one of the reasons for it. Mosaic augmentation [91] involves partitioning an image into four equal sections and subsequently merging them to create a new composite image, being first introduced on YOLOv4. This technique combines different portions of the original image to generate a transformed and augmented version, which can enhance the training dataset's diversity and variability.

Mosaic augmentation allows the network to become invariant to different image-capturing techniques, as a single image is composed of images taken with different altitudes and angles. Therefore, we decided to investigate this further to determine the precise impact of mosaic augmentation on the model performance and to ensure that its utilization aligns with our intended objectives. A comparison between batches with and without mosaic augmentation is shown in

Figure 7.2. This comparison highlights the impact of the mosaic augmentation in the training images.

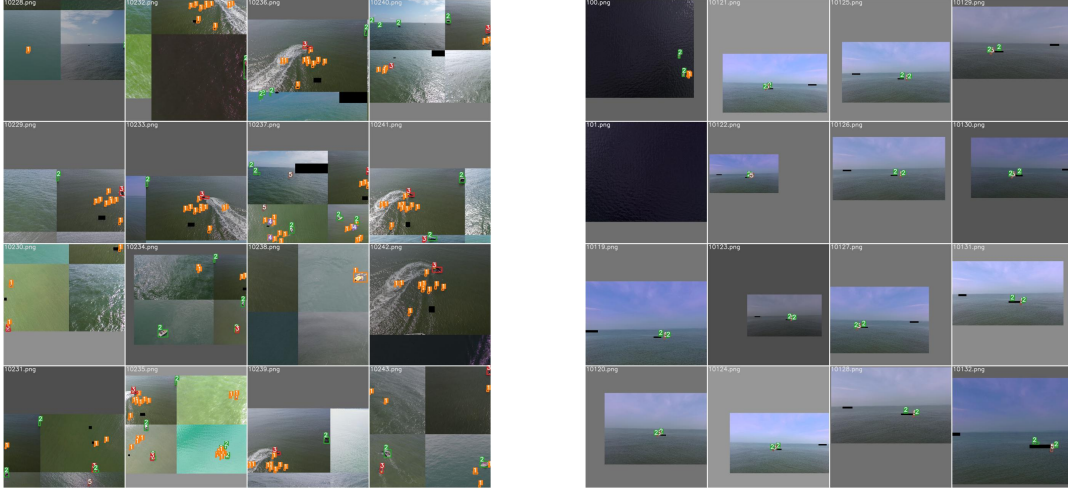


Figure 7.2: Comparison of batches with and without mosaic augmentation.

7.4.1 Baseline Without Mosaic Augmentation

To examine the influence of mosaic augmentation, we trained the model without employing this augmentation technique in order to have a baseline. We evaluated the performance of the baseline model using the evolved hyperparameters. It is also important to mention that without mosaic augmentation, the baseline model required 80 training epochs to converge without overfitting. In contrast, when mosaic augmentation was employed, the model was able to converge in only 20 epochs. Table 7.10 shows the achieved results on different seeds.

Table 7.10: Baseline performance without mosaic.

Model	mAP	mAP@50	mAP@75	AR1	AR10
Seed 0	33.73	57.52	34.09	29.49	39.67
Seed 42	35.51	60.34	35.96	31.79	41.77
Seed 821	35.50	59.07	36.17	31.29	41.24
Seed 1765	34.21	59.02	34.23	30.82	40.52
Average	34.74	58.99	35.11	30.85	40.80
Standard Deviation	0.91	1.15	1.10	0.99	0.91
Baseline Average using Mosaic	42.57	71.34	44.67	37.60	49.61
Difference	-7.83	-12.35	-9.56	-6.75	-8.81

The baseline without mosaic achieves an average mAP of 34.74%, which is 7.83 lower than the mAP achieved by the baseline using mosaic augmentation. These results show that this technique plays a crucial role in enhancing performance. The substantial gain in performance highlights the importance of using mosaic augmentation to enhance object detection performance in this dataset.

7.4.2 Domain Adversarial Learning Without Mosaic Augmentation

We performed more experiments to evaluate the performance of our architecture without the mosaic augmentation. We started by analyzing different combinations of the N and λ for the altitude domain. As can be seen in Table 7.11, the ones that gave the best results were $N = 8$ and a $\lambda = 0.005$ with an mAP of 35.95%.

Table 7.11: Performance on altitude domain without mosaic.

N	λ	mAP	mAP@50	mAP@75	AR1	AR10
8	0.001	35.36	60.88	36.07	31.59	41.79
	0.005	35.95	61.49	36.62	31.60	42.23
28	0.001	35.89	60.02	37.47	32.10	42.40
	0.005	35.36	60.88	36.07	31.57	41.78

We proceed to run the best combination in different seeds, obtaining the results presented in Table 7.12. In this scenario, we observed a higher increase in performance when compared to the performance gains using mosaic augmentation. In this case, the mAP had an improvement of 0.81 in relation to the baseline. One more time, the biggest increase in performance was at mAP@50, where the improvement was 1.76. The increase on mAP@75 was a little lower, with an improvement of 1.16.

Table 7.12: Performance using the best parameters regarding altitude without mosaic.

Model	mAP	mAP@50	mAP@75	AR1	AR10
Seed 0	35.95	61.49	36.62	31.6	42.23
Seed 42	35.59	60.88	36.47	32.00	42.73
Seed 821	35.58	60.94	36.10	31.52	41.99
Seed 1765	35.08	59.67	35.87	31.54	41.46
Average	35.55	60.75	36.27	31.67	42.10
Standard Deviation	0.36	0.77	0.34	0.23	0.53
Baseline Average	34.74	58.99	35.11	30.85	40.80
Improvement over Baseline Average	+0.81	+1.76	+1.16	+0.82	+1.30

The same experiments were performed on the angle variable, as shown in Table 7.13. The best values were $N = 8$ and $\lambda = 0.001$ which led to an mAP of 36.71%.

Table 7.13: Performance on angle domain without mosaic.

N	λ	mAP	mAP@50	mAP@75	AR1	AR10
8	0.001	36.71	62.26	37.73	32.76	43.18
	0.005	35.35	60.65	36.09	32.07	41.71
28	0.001	35.85	61.33	36.43	31.47	42.06
	0.005	36.2	60.97	37.09	32.38	42.55

Table 7.14 exhibits the outcomes obtained from multiple runs utilizing different seeds on the angle. The results show a 1.23 increase in mAP which is again a higher increase in the performance when compared to the results using mosaic. The biggest increase in performance was at mAP@50, where the improvement was 1.89, and the increase on mAP@75 was similarly high, with an improvement of 1.83.

Table 7.14: Performance using the best parameters regarding angle without mosaic.

Model	mAP	mAP@50	mAP@75	AR1	AR10
Seed 0	36.71	62.26	37.73	32.76	43.18
Seed 42	36.00	60.72	37.08	32.01	42.54
Seed 821	35.12	59.22	36.01	31.44	41.27
Seed 1765	36.05	61.33	36.92	32.10	42.21
Average	35.97	60.88	36.94	32.08	42.30
Standard Deviation	0.65	1.28	0.71	0.54	0.80
Baseline Average	34.74	58.99	35.11	30.85	40.80
Improvement over Baseline Average	+1.23	+1.89	+1.83	+1.23	+1.50

To further evaluate our model’s performance and robustness, we employed the multiple-domain adversarial. Using multiple domains, as depicted in Table 7.15, gave a lower performance than using a just single domain, similar to the results with the mosaic augmentation. For this reason, we have not performed the runs on different seeds.

Table 7.15: Performance using multiple domains without mosaic.

N	$\lambda_{Altitude}$	λ_{Angle}	mAP	mAP@50	mAP@75	AR1	AR10
8	0.001	0.001	36.37	61.99	36.73	32.28	42.85
	0.005	0.005	35.14	59.04	35.87	32.26	41.39

The ensemble approach is also identical to the one we performed before, and the results can be seen in Table 7.16. The model average was 2.11 better than the baseline when comparing the mAP and 3.47 when comparing mAP@50. However, the obtained results are still below the performance achieved when using the mosaic augmentation without integrating our architecture.

Table 7.16: Performance using ensemble model.

Model	mAP	mAP@50	mAP@75	AR1	AR10
Seed 0	37.19	62.35	38.08	32.77	43.95
Seed 42	36.77	62.75	37.65	32.95	44.20
Seed 821	36.56	61.97	37.42	32.49	43.20
Seed 1765	36.87	62.78	37.48	33.00	43.56
Average	36.85	62.46	37.66	32.80	43.73
Standard Deviation	0.26	0.38	0.30	0.23	0.44
Baseline Average	34.74	58.99	35.11	30.85	40.80
Improvement over Baseline Average	+2.11	+3.47	+2.55	+1.95	+2.93

These findings reinforce the significance of mosaic augmentation in this dataset. Mosaic augmentation aids the model in developing a more robust model, improving its performance. The slightly higher results of our approach when not using the mosaic augmentation show that it also helps to make the network more invariant to changes in the angle and altitude. It should also be noted that the use of non-orthogonal domains may influence these results, and further tests should be conducted on datasets with orthogonal metadata to draw a more comprehensive understanding regarding the merits of one technique relative to the other.

7.5 Hypothesis and Research Questions Discussion

The hypothesis formulated in Section 4.2 was the following:

There is a way to improve the model's ability to detect objects in images from maritime environments without or with minimal overhead in complexity by resorting to metadata gathered from UAV flight.

In this chapter, we saw that it is possible to use metadata to improve performance without increasing the complexity of the underlying neural network, indicating that the hypothesis is possibly true. The answers to the proposed research questions that helped us reach this conclusion are the following:

RQ1. Can the performance of maritime object detection neural networks be improved by incorporating metadata containing flight information?

We found out that our model when using metadata, performs better than the ones without metadata, even though the gains are low. However, it is essential to consider that the metadata used in our study was non-orthogonal, which may have contributed to the low gains. To further enhance performance, it would be beneficial to utilize diverse metadata encompassing a range of factors, such as adverse weather conditions, which will allow us to define orthogonal domains and potentially achieve a more substantial increase in performance.

RQ2. Can we incorporate metadata containing flight information without increasing the neural network complexity?

In our proposed methodology, we have successfully integrated the metadata containing flight information into YOLOv7, a state-of-the-art object detection network. It is important to note that our methodology ensures that the incorporation of metadata does not lead to an increase in the complexity of the neural network. By carefully designing the integration process, we have ensured that the existing architecture and computational requirements of YOLOv7 remain unchanged while incorporating the flight information metadata. However, when using the approach of model ensemble, the complexity is increased, but not due to our metadata integration.

RQ3. Would it be possible to apply our methodology to integrate metadata easily on other object detection networks?

Our methodology can be applied to integrate metadata into various object detection networks with relative ease. The flexibility of our approach allows for the seamless integration of metadata into any arbitrary neural network architecture by adapting the training loop and incorporating the domain classification module. This demonstrates the versatility and potential applicability of our approach beyond the YOLOv7.

7.6 Summary

We first started this chapter by explaining in Section 7.1 what we did to make the results more reproducible by using PyTorch guidelines on reproducible runs. We establish our baseline in Section 7.2 by performing a hyperparameter tuning with an evolutionary algorithm over 350 iterations on top of the default hyperparameters. Section 7.3 provided an analysis of the performance of our architecture. We began by individually assessing the impact of angle and altitude adversarial training. However, the gains achieved from these techniques were relatively low, providing only marginal performance improvements. Next, we examined the combined effect of both angle adversarial and altitude adversarial training, and the performance was lower than training each domain separately. The reason for this could be justified because the angle and altitude are not orthogonal. Furthermore, we explored the effectiveness of an ensemble model, where the results showed a higher increase in performance. A summary of these results can be found in Table 7.17. Furthermore, we investigate the impact of mosaic augmentation on the performance of our network in Section 7.4. This augmentation technique, by merging four images together, helps to enhance the robustness of the network in enhancing performance and aiding in better generalization. The results showed a decrease compared to the previous outcomes. Nevertheless, our methodology exhibited slightly greater improvements in performance when compared to the gains of our architecture using mosaic augmentation. The results of the experiments without mosaic augmentation are presented in Table 7.18. After assessing all the results, we discuss the previously formulated hypothesis and the research questions in Section 7.5.

Table 7.17: Average performance of each model.

Model	mAP	mAP@50	mAP@75	AR1	AR10
Baseline with Default Hyperparameters	42.20	71.95	43.55	38.30	50.88
Baseline with Evolved Hyperparameters	42.57	71.34	44.67	37.60	49.61
Single Domain with Altitude	43.05	72.75	45.04	37.71	50.34
Single Domain with Angle	43.24	72.64	45.14	38.14	50.54
Multi-Domain with Altitude and Angle	42.88	71.99	44.98	37.50	49.89
Ensemble with Altitude and Angle	44.01	74.60	45.72	38.82	51.85

Table 7.18: Average performance of each model without mosaic.

Model	mAP	mAP@50	mAP@75	AR1	AR10
Baseline	34.74	58.99	35.11	30.85	40.8
Single Domain with Altitude Without Mosaic	35.55	60.75	36.27	31.67	42.10
Single Domain with Angle Without Mosaic	35.97	60.88	36.94	32.08	42.30
Ensemble with Altitude and Angle	36.85	62.46	37.66	32.80	43.73

Chapter 8

Conclusion

This chapter summarizes the conclusions and results drawn from the current work. Section 8.1 provides a comprehensive overview of these conclusions, highlighting the essential findings and insights gained throughout the study. Additionally, the contributions made by this research are presented in Section 8.2, outlining the novel aspects and advancements introduced in this field of study. Finally, Section 8.3 delves into the discussion of future work, identifying potential directions for further research and development to build upon the findings of this work.

8.1 Conclusions

The use of UAVs for maritime surveillance is increasing, suggesting the importance of maritime object detection. Efficient algorithms capable of detecting objects with high precision and accuracy are reducing the necessity to look through videos and images gathered on oceans manually. Furthermore, UAVs are also used in other surveillance tasks, making it possible to extend this work to other areas in the future.

To review the state of the art, a methodological review was performed guided by the literature research questions. The results from this review revealed some open challenges. Most of the existing techniques regarding object detection used attention modules, multi-scale detectors, and anchor boxes, which helped to improve precision and recall. Despite the availability of flight metadata from UAVs, our investigation revealed that these data were not being utilized in object detection systems.

After the dataset analysis, we found out that the relevant metadata was the angle and altitude of the UAV, which have a direct impact on the characteristics of the captured objects. Acknowledging this, we designed a robust network for these image-capturing conditions. Our architecture was built on top of the YOLOv7, a state-of-the-art real-time object detector, and for training, we employed an adversarial learning approach to learn invariant features across multiple domains. The metadata was solely used during the training process, allowing the algorithm to be efficient at inference. We also designed an ensemble method combining multiple models, which is more complex and less suitable for deployment on edge devices.

Comprehensive experiments and analyses were conducted to evaluate this methodology's effectiveness. The gains achieved over the baseline were minimal when utilizing a single domain. Surprisingly, when incorporating the adversarial domain alongside the other domain simultaneously, the gains were lower compared to using just a single domain. One potential reason for lower performance when utilizing multiple domains is that they may not be orthogonal in terms of their impact on the captured images. As a result, during network training, the gradients generated during backpropagation may cancel each other. On the other hand, the ensemble model merging two models gave better results, having a higher improvement than the two previous methodologies.

We also theorize that the mosaic augmentation would also increase the robustness of our neural network. We discovered that this augmentation helped the model to have a better performance, being fundamental in this dataset. We also found that the gains using our approach without mosaic were higher, showing that mosaic also helps the network to become invariant to these variances.

Further experiments should be conducted using datasets that offer more diverse metadata. Also, to capture more real maritime surveillance scenarios, it is crucial to include representative conditions encompassing adverse weather and challenging sea conditions within the dataset, enabling the establishment of orthogonal domains. Unfortunately, current publicly available datasets don't have images with adverse conditions along with metadata, as including them could provide further insights into the merits of this adversarial model training performance.

8.2 Contributions

This work made significant scientific contributions by introducing novel methodologies and addressing specific research problems. The main contributions were:

Literature review regarding maritime surveillance

A literature review of the different approaches employed to detect maritime objects was discussed and analyzed throughout this work. This literature review includes the available datasets, which techniques are used for object detection to deal with particular challenges of this environment (such as small objects or extreme glare), and optimization methods to enable these algorithms to perform real-time detection.

Efficient neural network methodology that incorporates flight metadata from UAVs

We integrated metadata from UAV platforms in neural networks to improve performance. At the same time, our methodology is efficient since the metadata is only used at training time, not affecting the inference performance. The computational complexity of our ensemble model is higher since it merges two models together.

8.3 Future Work

The solution developed during this thesis represents a preliminary effort to use metadata from UAVs in the context of maritime surveillance. While this work has provided valuable insights, several avenues remain for further exploration and utilization of the metadata's full potential. These include:

- **Temporal Correlation:** Using temporal correlation of the data could be better to handle and track objects in real-time.
- **Specialized Network:** Create an ensemble network that consists of multiple subnetworks where each specializes in detecting a specific category of altitude and angle images. For example, three networks where each one specializes in detecting high, medium, and low altitude images. By combining the outputs of these specialized subnetworks, the ensemble network can provide enhanced performance and predictive capabilities for both altitude and angle classification in the context of maritime surveillance.
- **Merging Metadata with Feature Maps:** Rather than relying on metadata to create a more invariant network to the given domains, an alternative approach is to directly integrate the metadata into the feature maps, similar to the approach from Sun et al. [33]. With this approach, the metadata is used both at training and inference time.
- **Data Gathering:** Datasets that contain metadata in the context of maritime surveillance are scarce, as highlighted in the literature review. Increasing the quantity and diversity of data can help train more robust models.

In conclusion, while this dissertation has laid the groundwork for using metadata in UAV-based maritime surveillance, there is a need for further research and development regarding its integration. The proposed future work provides insights and a roadmap for advancing the utilization of metadata in maritime surveillance.

References

- [1] Ashish Bhandari, Anil Kumar, and GK Singh. Cuckoo search algorithm based satellite image contrast and brightness enhancement using DWT-SVD. *ISA Transactions*, 53:1286–1296, June 2014.
- [2] Panagiotis Barmpoutis, Periklis Papaioannou, Kosmas Dimitropoulos, and Nikos Grammalidis. A Review on Early Forest Fire Detection Systems Using Optical Remote Sensing. *Sensors*, 20(22):6442, January 2020.
- [3] Zongyao Yang, Xueying Yu, Simon Dedman, Massimiliano Rosso, Jingmin Zhu, Jiaqi Yang, Yuxiang Xia, Yichao Tian, Guangping Zhang, and Jingzhen Wang. UAV remote sensing applications in marine monitoring: Knowledge visualization and review. *Science of The Total Environment*, 838:155939, September 2022.
- [4] Xi Liang, Jing Zhang, Li Zhuo, Yuzhao Li, and Qi Tian. Small Object Detection in Unmanned Aerial Vehicle Images Using Feature Fusion and Scaling-Based Single Shot Detector With Spatial Context Analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(6):1758–1770, June 2020.
- [5] Ricardo Ribeiro, Goncalo Cruz, Jorge Matos, and Alexandre Bernardino. A Data Set for Airborne Maritime Surveillance Environments. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(9):2720–2732, September 2019.
- [6] Wenli Sun and Xu Gao. Object Detection in Maritime Scenarios Based on Swin-Transformer. In Yuriy S. Shmaliy and Abdelhalim Abdelnaby Zekry, editors, *6th International Technical Conference on Advances in Computing, Control and Industrial Engineering (CCIE 2021)*, Lecture Notes in Electrical Engineering, pages 786–798, Singapore, 2022. Springer Nature.
- [7] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection, May 2016.
- [8] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, volume 1, pages 886–893, San Diego, CA, USA, 2005. IEEE.
- [9] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, Lecture Notes in Computer Science, pages 404–417, Berlin, Heidelberg, 2006. Springer.
- [10] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. Object Detection With Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, November 2019.

- [11] Magda Alexandra Trujillo-Jiménez. Summarization of video from Feature Extraction Method using Image Processing and Artificial Intelligence. January 2018.
- [12] Sung-Sik Park, Van-Than Tran, and Dong-Eun Lee. Application of Various YOLO Models for Computer Vision-Based Real-Time Pothole Detection. *Applied Sciences*, 11(23):11229, January 2021.
- [13] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, Honolulu, HI, July 2017. IEEE.
- [14] Manuel Carranza-García, Jesús Torres-Mateo, Pedro Lara-Benítez, and Jorge García-Gutiérrez. On the Performance of One-Stage and Two-Stage Object Detectors in Autonomous Vehicles Using Camera Data. *Remote Sensing*, 13(1):89, January 2021.
- [15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, January 2016.
- [16] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement, April 2018.
- [17] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, July 2022.
- [18] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. YOLO by Ultralytics, 1 2023.
- [19] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. volume 9905, pages 21–37. 2016.
- [20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection, February 2018.
- [21] Meng-Hao Guo, Tian-Xing Xu, Jiang-Jiang Liu, Zheng-Ning Liu, Peng-Tao Jiang, Tai-Jiang Mu, Song-Hai Zhang, Ralph R. Martin, Ming-Ming Cheng, and Shi-Min Hu. Attention Mechanisms in Computer Vision: A Survey. *Computational Visual Media*, 8(3):331–368, September 2022.
- [22] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: Convolutional Block Attention Module, July 2018.
- [23] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-Excitation Networks, May 2019.
- [24] Bo-Kai Ruan, Hong-Han Shuai, and Wen-Huang Cheng. Vision Transformers: State of the Art and Research Challenges, July 2022.
- [25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, December 2017.
- [26] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, June 2021.

- [27] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, August 2021.
- [28] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers, May 2020.
- [29] Rafael Padilla, Sergio L. Netto, and Eduardo A. B. da Silva. A Survey on Performance Metrics for Object-Detection Algorithms. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 237–242, July 2020.
- [30] Sam Fletcher and Md Zahidul Islam. Comparing sets of patterns with the Jaccard index. *Australasian Journal of Information Systems*, 22, March 2018.
- [31] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context, February 2015.
- [32] Jan Hosang, Rodrigo Benenson, Piotr Dollár, and Bernt Schiele. What Makes for Effective Detection Proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):814–830, April 2016.
- [33] Qilin Sun, Chao Huang, Minjie Chen, Hui Xu, and Yali Yang. Skin Lesion Classification Using Additional Patient Information. *BioMed Research International*, 2021:1–6, April 2021.
- [34] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R. Arabnia. A Brief Review of Domain Adaptation, October 2020.
- [35] Yaroslav Ganin and Victor Lempitsky. Unsupervised Domain Adaptation by Backpropagation, February 2015.
- [36] Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Multi-Adversarial Domain Adaptation, September 2018.
- [37] Leonardo Capozzi, Vítor Barbosa, Carolina Pinto, João Ribeiro Pinto, Américo Pereira, Pedro M. Carvalho, and Jaime S. Cardoso. Toward Vehicle Occupant-Invariant Models for Activity Characterization. *IEEE Access*, 10:104215–104225, 2022.
- [38] Jinlong Li, Runsheng Xu, Jin Ma, Qin Zou, Jiaqi Ma, and Hongkai Yu. Domain Adaptive Object Detection for Autonomous Driving under Foggy Weather, October 2022.
- [39] Pedro M. Ferreira, Diogo Pernes, Ana Rebelo, and Jaime S. Cardoso. Learning signer-invariant representations with adversarial training. In Wolfgang Osten and Dmitry P. Nikolaev, editors, *Twelfth International Conference on Machine Vision (ICMV 2019)*, page 117, Amsterdam, Netherlands, January 2020. SPIE.
- [40] Zhenyu Wu, Karthik Suresh, Priya Narayanan, Hongyu Xu, Heesung Kwon, and Zhangyang Wang. Delving Into Robust Object Detection From Unmanned Aerial Vehicles: A Deep Nuisance Disentanglement Approach. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1201–1210, Seoul, Korea (South), October 2019. IEEE.
- [41] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep Domain Confusion: Maximizing for Domain Invariance, December 2014.

- [42] Eduardo Teixeira, Beatriz Araujo, Victor Costa, Samuel Mafra, and Felipe Figueiredo. Literature Review on Ship Localization, Classification, and Detection Methods Based on Optical Sensors and Neural Networks. *Sensors*, 22(18):6879, January 2022.
- [43] Shunjun Wei, Xiangfeng Zeng, Qizhe Qu, Mou Wang, Hao Su, and Jun Shi. HRSID: A High-Resolution SAR Images Dataset for Ship Detection and Instance Segmentation. *IEEE Access*, 8:120234–120254, 2020.
- [44] Leon Amadeus Varga, Benjamin Kiefer, Martin Messmer, and Andreas Zell. SeaDronesSee: A Maritime Benchmark for Detecting Humans in Open Water. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3686–3696, Waikoloa, HI, USA, January 2022. IEEE.
- [45] Peng Chen, Ying Li, Hui Zhou, Bingxin Liu, and Peng Liu. Detection of Small Ship Objects Using Anchor Boxes Cluster and Feature Pyramid Network Model for SAR Imagery. *Journal of Marine Science and Engineering*, 8(2):112, February 2020.
- [46] Urška Kanjir, Harm Greidanus, and Krištof Oštir. Vessel detection and classification from spaceborne optical images: A literature survey. *Remote Sensing of Environment*, 207:1–26, March 2018.
- [47] Zeyuan Shao, Hongguang Lyu, Yong Yin, Tao Cheng, Xiaowei Gao, Wenjun Zhang, Qianfeng Jing, Yanjie Zhao, and Lunping Zhang. Multi-Scale Object Detection Model for Autonomous Ship Navigation in Maritime Environment. *Journal of Marine Science and Engineering*, 10(11):1783, November 2022.
- [48] Tianwen Zhang, Xiaoling Zhang, Jianwei Li, Xiaowo Xu, Baoyou Wang, Xu Zhan, Yanqin Xu, Xiao Ke, Tianjiao Zeng, Hao Su, Israr Ahmad, Dece Pan, Chang Liu, Yue Zhou, Jun Shi, and Shunjun Wei. SAR Ship Detection Dataset (SSDD): Official Release and Comprehensive Data Analysis. *Remote Sensing*, 13(18):3690, January 2021.
- [49] Airbus ship detection challenge. <https://www.kaggle.com/c/airbus-ship-detection>, 2018. [Accessed 05-Feb-2023].
- [50] Zhenfeng Shao, Wenjing Wu, Zhongyuan Wang, Wan Du, and Chengyuan Li. SeaShips: A Large-Scale Precisely Annotated Dataset for Ship Detection. *IEEE Transactions on Multimedia*, 20(10):2593–2604, October 2018.
- [51] Abhilasha Nanda, Sung Won Cho, Hyeopwoo Lee, and Jin Hyoung Park. KOLOMVERSE: KRISO open large-scale image dataset for object detection in the maritime universe, June 2022.
- [52] Dilip K. Prasad, Deepu Rajan, Lily Rachmawati, Eshan Rajabally, and Chai Quek. Video Processing From Electro-Optical Sensors for Object Detection and Tracking in a Maritime Environment: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, 18(8):1993–2016, August 2017.
- [53] Zhiguo Zhou, Jiaen Sun, Jiabao Yu, Kaiyuan Liu, Junwei Duan, Long Chen, and C. L. Philip Chen. An Image-Based Benchmark Dataset and a Novel Object Detector for Water Surface Object Detection. *Frontiers in Neurorobotics*, 15, 2021.

- [54] Xinqiang Chen, Lei Qi, Yongsheng Yang, Qiang Luo, Octavian Postolache, Jinjun Tang, and Huafeng Wu. Video-Based Detection Infrastructure Enhancement for Automated Ship Recognition and Behavior Analysis. *Journal of Advanced Transportation*, 2020:1–12, January 2020.
- [55] 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, March 2022.
- [56] Benjamin Kiefer, David Ott, and Andreas Zell. Leveraging Synthetic Data in Object Detection on Unmanned Aerial Vehicles, December 2021.
- [57] Gonalo Cruz and Alexandre Bernardino. Aerial Detection in Maritime Scenarios Using Convolutional Neural Networks. In Jacques Blanc-Talon, Cosimo Distanto, Wilfried Philips, Dan Popescu, and Paul Scheunders, editors, *Advanced Concepts for Intelligent Vision Systems*, Lecture Notes in Computer Science, pages 373–384, Cham, 2016. Springer International Publishing.
- [58] Sung-Jun Lee, Myung-II Roh, and Min-Jae Oh. Image-based ship detection using deep learning. *Ocean Systems Engineering*, 10(4):415–434, December 2020.
- [59] Ming Liu, Linbo Tang, and Zongya Li. Real-Time Object Detection in UAV Vision based on Neural Processing Units. In *2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC)*, pages 1951–1955, Chongqing, China, March 2022. IEEE.
- [60] Yihong Zhang, Hang Ge, Qin Lin, Ming Zhang, and Qiantao Sun. Research of Maritime Object Detection Method in Foggy Environment Based on Improved Model SRC-YOLO. *Sensors*, 22(20):7786, January 2022.
- [61] D.J. Jobson, Z. Rahman, and G.A. Woodell. Properties and performance of a center/surround retinex. *IEEE Transactions on Image Processing*, 6(3):451–462, March 1997.
- [62] Xin Nie, Meifang Yang, and Ryan Wen Liu. Deep Neural Network-Based Robust Ship Detection Under Different Weather Conditions. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 47–52, Auckland, New Zealand, October 2019. IEEE.
- [63] Ryan Wen Liu, Weiqiao Yuan, Xinqiang Chen, and Yuxu Lu. An enhanced CNN-enabled learning method for promoting ship detection in maritime surveillance system. *Ocean Engineering*, 235:109435, September 2021.
- [64] Zhijun Chen, Depeng Chen, Yishi Zhang, Xiaozhao Cheng, Mingyang Zhang, and Chaozhong Wu. Deep learning for autonomous ship-oriented small ship detection. *Safety Science*, 130:104812, October 2020.
- [65] Hangyue Zhao, Hongpu Zhang, and Yanyun Zhao. YOLOv7-Sea: Object Detection of Maritime UAV Images Based on Improved YOLOv7.
- [66] Fatih Cagatay Akyon, Sinan Onur Altinuc, and Alptekin Temizel. Slicing Aided Hyper Inference and Fine-tuning for Small Object Detection. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 966–970, October 2022.
- [67] Lei Yang, Guowu Yuan, Hao Zhou, Hongyu Liu, Jian Chen, and Hao Wu. RS-YOLOX: A High-Precision Detector for Object Detection in Satellite Remote Sensing Images. *Applied Sciences*, 12(17):8707, January 2022.

- [68] Xin Wang, Ning He, Chen Hong, Qi Wang, and Ming Chen. Improved YOLOX-X based UAV aerial photography object detection algorithm. Preprint, In Review, October 2022.
- [69] Qingpeng Li, Lichao Mou, Qingjie Liu, Yunhong Wang, and Xiao Xiang Zhu. HSF-Net: Multiscale Deep Feature Embedding for Ship Detection in Optical Remote Sensing Imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 56(12):7147–7161, December 2018.
- [70] Yulian Zhang, Lihong Guo, Zengfa Wang, Yang Yu, Xinwei Liu, and Fang Xu. Intelligent Ship Detection in Remote Sensing Images Based on Multi-Layer Convolutional Feature Fusion. *Remote Sensing*, 12(20):3316, January 2020.
- [71] Tao Liu, Bo Pang, Shangmao Ai, and Xiaoqiang Sun. Study on Visual Detection Algorithm of Sea Surface Targets Based on Improved YOLOv3. *Sensors*, 20(24):7263, January 2020.
- [72] Zhonghua Hong, Ting Yang, Xiaohua Tong, Yun Zhang, Shenlu Jiang, Ruyan Zhou, Yanling Han, Jing Wang, Shuhu Yang, and Sichong Liu. Multi-Scale Ship Detection From SAR and Optical Imagery Via A More Accurate YOLOv3. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:6083–6101, 2021.
- [73] Hao Li, Lianbing Deng, Cheng Yang, Jianbo Liu, and Zhaoquan Gu. Enhanced YOLO v3 Tiny Network for Real-Time Ship Detection From Visual Image. *IEEE Access*, 9:16692–16706, 2021.
- [74] 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(4):660, February 2021.
- [75] Ruixin Ma, Kexin Bao, and Yong Yin. Improved Ship Object Detection in Low-Illumination Environments Using RetinaMFANet. *Journal of Marine Science and Engineering*, 10(12):1996, December 2022.
- [76] Yongshuai Li, Haiwen Yuan, Yuan Wang, and Bulin Zhang. Maritime Vessel Detection and Tracking under UAV Vision. In *2022 International Conference on Artificial Intelligence and Computer Information Technology (AICIT)*, pages 1–4, September 2022.
- [77] Yongshuai Li, Haiwen Yuan, Yanfeng Wang, and Changshi Xiao. GGT-YOLO: A Novel Object Detection Algorithm for Drone-Based Maritime Cruising. *Drones*, 6(11):335, November 2022.
- [78] Haibin Duan, Xiaobin Xu, Yimin Deng, and Zhigang Zeng. Unmanned Aerial Vehicle Recognition of Maritime Small-Target Based on Biological Eagle-Eye Vision Adaptation Mechanism. *IEEE Transactions on Aerospace and Electronic Systems*, 57(5):3368–3382, October 2021.
- [79] Xue Yang, Hao Sun, Xian Sun, Menglong Yan, Zhi Guo, and Kun Fu. Position Detection and Direction Prediction for Arbitrary-Oriented Ships via Multitask Rotation Region Convolutional Neural Network. *IEEE Access*, 6:50839–50849, 2018.
- [80] Xue Yang, Jirui Yang, Junchi Yan, Yue Zhang, Tengfei Zhang, Zhi Guo, Sun Xian, and Kun Fu. SCRDet: Towards More Robust Detection for Small, Cluttered and Rotated Objects, August 2019.

- [81] Zikang Shao, Xiaoling Zhang, Tianwen Zhang, Xiaowo Xu, and Tianjiao Zeng. RBFA-Net: A Rotated Balanced Feature-Aligned Network for Rotated SAR Ship Detection and Classification. *Remote Sensing*, 14(14):3345, January 2022.
- [82] Goncalo Cruz and Alexandre Bernardino. Learning Temporal Features for Detection on Maritime Airborne Video Sequences Using Convolutional LSTM. *IEEE Transactions on Geoscience and Remote Sensing*, 57(9):6565–6576, September 2019.
- [83] Tao Liu, Bo Pang, Lei Zhang, Wei Yang, and Xiaoqiang Sun. Sea Surface Object Detection Algorithm Based on YOLO v4 Fused with Reverse Depthwise Separable Convolution (RDSC) for USV. *Journal of Marine Science and Engineering*, 9(7):753, July 2021.
- [84] Zhangqi Yang, Ye Li, Bo Wang, Shuoshuo Ding, and Peng Jiang. A Lightweight Sea Surface Object Detection Network for Unmanned Surface Vehicles. *Journal of Marine Science and Engineering*, 10(7):965, July 2022.
- [85] Mrunalini Nalamati, Nabin Sharma, Muhammad Saqib, and Michael Blumenstein. Automated Monitoring in Maritime Video Surveillance System. In *2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6, November 2020.
- [86] J Sharafaldeen, M Rizk, D Heller, A Baghdadi, and J-Ph Diguët. Marine Object Detection Based on Top-View Scenes Using Deep Learning on Edge Devices.
- [87] Xiaowo Xu, Xiaoling Zhang, and Tianwen Zhang. Lite-YOLOv5: A Lightweight Deep Learning Detector for On-Board Ship Detection in Large-Scene Sentinel-1 SAR Images. *Remote Sensing*, 14(4):1018, January 2022.
- [88] M.V. Zelkowitz and D.R. Wallace. Experimental models for validating technology. *Computer*, 31(5):23–31, May 1998.
- [89] Benjamin Kiefer, Matej Kristan, Janez Pers, Lojze Zust, Fabio Poiesi, Fabio Andrade, Alexandre Bernardino, Matthew Dawkins, Jenni Raitoharju, Yitong Quan, Adem Atmaca, Timon Hofer, Qiming Zhang, Yufei Xu, Jing Zhang, Dacheng Tao, Lars Sommer, Raphael Spraul, Hangyue Zhao, Hongpu Zhang, Yanyun Zhao, Jan Lukas Augustin, Eui-ik Jeon, Impyeong Lee, Luca Zedda, Andrea Loddo, Cecilia Di Ruberto, Sagar Verma, Siddharth Gupta, Shishir Muralidhara, Niharika Hegde, Daitao Xing, Nikolaos Evangeliou, Anthony Tzes, Vojtech Bartl, Jakub Spanhel, Adam Herout, Neelanjan Bhowmik, Toby P Breckon, Shivanand Kundargi, Tejas Anvekar, Ramesh Ashok Tabib, Uma Mudenagudi, Arpita Vats, Yang Song, DeLong Liu, Yonglin Li, Shuman Li, Chenhao Tan, Long Lan, Vladimir Somers, Christophe De Vleeschouwer, Alexandre Alahi, Hsiang-Wei Huang, Cheng-Yen Yang, Jenq-Neng Hwang, Pyong-Kun Kim, Kwangju Kim, Kyoungoh Lee, Shuai Jiang, Haiwen Li, Zheng Ziqiang, Tuan-Anh Vu, Hai Nguyen-Truong, Sai-Kit Yeung, Zhuang Jia, Sophia Yang, Chih-Chung Hsu, Xiu-Yu Hou, Yu-An Jhang, Simon Yang, and Mau-Tsuen Yang. 1st Workshop on Maritime Computer Vision (MaCVi) 2023: Challenge Results.
- [90] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization.
- [91] Parvinder Kaur, Baljit Singh Khehra, and Er. Bhupinder Singh Mavi. Data Augmentation for Object Detection: A Review. In *2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 537–543, August 2021.

Appendix A

Data Annotations

A.1 Object Detection Format

Each entry in the JSON file corresponds to a specific object instance within an image, containing the bounding box coordinates ("bbox") which represent the object location, the area of the bounding box ("area"), and the category ("category_id") indicating the object's class or category. One snippet of the JSON file can be seen in the following listing.

```
1 {  
2     "id": 16247,  
3     "image_id": 3835,  
4     "bbox": [3392, 1024, 24, 30],  
5     "area": 720,  
6     "category_id": 2  
7 }, {  
8     "id": 16248,  
9     "image_id": 3835,  
10    "bbox": [1872, 1139, 61, 27],  
11    "area": 1647,  
12    "category_id": 2  
13 }
```

A.2 Metadata Format

The available metadata provides not only general details such as the image name, size, and date of capture but also specific information about the device used to capture the photo and the corresponding frame number from the video.

Additionally, it contains more detailed and specific information under the "meta" key. This metadata offers insights into various drone-related attributes, which can include parameters such as altitude, camera angle, and GPS coordinates, among other information.

```
1 {
2   "id": 3835,
3   "file_name": "3835.png",
4   "height": 2160,
5   "width": 3840,
6   "source": {
7     "drone": "mavic",
8     "folder_name": "NorthSea_Fri_DJI_0110",
9     "video": "NorthSea_Fri_DJI_0110.MP4",
10    "frame_no": 1380
11  },
12  "date_time": "2021-08-20T14:55:43.012633",
13  "meta": {
14    "image_name": "DJI_0110.MP4_image1330.png",
15    "datetime(utc)": "2021-08-20 12:55:43",
16    "latitude": 53.7202368189897,
17    "longitude": 7.168472087897331,
18    "height_above_takeoff(meter)": 16.999170984455958,
19    "speed(m/s)": 11.42969342869915,
20    "xspeed(m/s)": 0.7999785426911041,
21    "yspeed(m/s)": -11.399694233348233,
22    "zspeed(m/s)": -0.19999463567277603,
23    "compass_heading(degrees)": 280.8,
24    "gimbal_heading(degrees)": 271.4,
25    "gimbal_pitch(degrees)": 1.6
26  }
27 }
```


Appendix B

Hyperparameters

Table B.1: Hyperparameters after evolution over 350 iterations.

Name	Value
lr0	0.00843
lrf	0.145
momentum	0.941
weight_decay	0.00036
warmup_epochs	2.12
warmup_momentum	0.822
warmup_bias_lr	0.0876
box	0.0513
cls	0.319
cls_pw	0.833
obj	0.628
obj_pw	0.764
iou_t	0.2
anchor_t	5.14
fl_gamma	0.0
hsv_h	0.015
hsv_s	0.512
hsv_v	0.33
degrees	0.0
translate	0.22
scale	0.461
shear	0.0
perspective	0.0
flipud	0.0
fliplr	0.408
mosaic	1.0
mixup	0.0
copy_paste	0.0
paste_in	0.0
loss_ota	0.696
anchors	3.05

