UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Master´s Degree in Deep Learning for
Audio and Video Signal Processing

MASTER THESIS

FOUNDATION MODEL-BASED SHIP DETECTION IN SATELLITE IMAGERY FOR MARITIME SURVEILLANCE

Alberto Espinosa Pérez
Advisor: Paula Martí Rocafull
Lecturer: Jesús Bescós Cano

JUNE 2025

FOUNDATION MODEL-BASED SHIP DETECTION IN SATELLITE IMAGERY FOR MARITIME SURVEILLANCE

Alberto Espinosa Pérez
Advisor: Paula Martí Rocafull
Lecturer: Jesús Bescós Cano

Dpto. Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
JUNE 2025

Resumen

El presente Trabajo Fin de Máster aborda la detección automática de embarcaciones en imágenes de satélite de resolución media (PlanetScope, 3 m) mediante un enfoque basado en *foundation models*. Se propone una estrategia de dos fases: (i) un pre-entrenamiento auto-supervisado a gran escala, empleando una versión mejorada de la arquitectura U-Net sobre $\approx$ 60 000 parches no etiquetados de 6 bandas espectrales, y (ii) un ajuste fino supervisado con únicamente 280 parches anotados que contienen buques. El objetivo es reducir drásticamente la necesidad de datos etiquetados y, al mismo tiempo, alcanzar una precisión operativa acorde con los exigentes requisitos de vigilancia marítima.

El modelo pre-entrenado captura representaciones robustas gracias a la reconstrucción de regiones enmascaradas, mientras que el ajuste fino optimiza una pérdida híbrida BCE + IoU específicamente orientada a objetos pequeños y escenarios con desequilibrio de clases extremo (agua $\gg$ buque). El sistema resultante obtiene un **IoU por instancia del 0,536 y un F1 del 0,671**, reduciendo los falsos positivos a 0,04% de los píxeles de agua y teniendo una tasa de fallo baja. Estos resultados superan en +0,155 IoU y +0,309 F1 a la mejor línea base entrenada desde cero.

La metodología se integra en el proyecto **MARVISION**, financiado por CDTI y fondos Next Generation EU, cuyo fin es validar un prototipo de vigilancia marítima para el Ministerio de Defensa. El modelo alcanza un rendimiento de inferencia de $\sim$16 parches/50 ms en una GPU A40, cumpliendo los requisitos de tiempo casi real del pipeline MARVISION. La tesis demuestra así que los modelos fundacionales permiten obtener segmentación fiable de barcos con una fracción del coste de anotación tradicional y sientan las bases para futuras extensiones multirresolución y multi-sensor.

Palabras clave

modelos fundacionales, aprendizaje auto-supervisado, detección de barcos, PlanetScope, segmentación semántica, vigilancia marítima

Abstract

This thesis addresses automatic ship detection in medium-resolution satellite imagery (PlanetScope, 3 m) through a foundation-model approach. The proposed pipeline consists of two stages: (i) large-scale self-supervised pre-training of an enhanced U-Net on ∼60 000 unlabeled six-band patches; and (ii) supervised fine-tuning with only 280 labeled vessel-containing patches. The goal is to drastically reduce annotation cost while achieving operational-grade accuracy for maritime-surveillance applications.

During pre-training, the network learns robust representations by reconstructing randomly masked regions. Fine-tuning employs a hybrid BCE + IoU loss, tailored to small objects and severe class imbalance (water ≫ vessel). The resulting model delivers an **instance-wise IoU of 0.536 and an instance F1-score of 0.671**, lowering false positives to 0.04% of water pixels and having a low error rate in detection. Compared with the best baseline trained from scratch, this corresponds to gains of +0.155 IoU and +0.309 F1.

The methodology is deployed within **MARVISION**, a CDTI-funded project supported by Next Generation EU funds and validated by the Spanish Ministry of Defense. Batch inference processes ≈16 six-band patches per 50 ms on a single NVIDIA A40 GPU, fulfilling near-real-time requirements of the MARVISION processing chain. The study demonstrates that foundation models enable reliable ship segmentation with an order-of-magnitude fewer labels and establishes a basis for forthcoming multi-resolution and multi-sensor extensions.

Keywords

foundation models, self-supervised learning, ship detection, PlanetScope, semantic segmentation, maritime surveillance

Acknowledgments

The MARVISION project has been awarded by the **Spanish Center for Technological Development and Innovation (CDTI)**, with the Ministry of Defense as the end user through the **General Directorate of Armament and Material (DGAM/PLATIN)**, and funded with European funds from the **Recovery and Resilience Mechanism (MRR)** "Next Generation EU". The objective of this contract is the development of a prototype operational system for maritime surveillance and its validation by the Ministry of Defense. This project contributes to the **Aerospace PERTE** in its **Action ACT9 – Spanish Earth Observation System for Security and Defense**, specifically addressing Technological Challenge No. 2.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Maritime surveillance is a critical application of Earth observation, with satellite imagery offering wide-area monitoring of vessel traffic for purposes such as fisheries enforcement, pollution control and border security [2]. Detecting ships from space, however, poses significant challenges. Targets tend to be very small relative to the image scale and can appear as only a few pixels, especially in medium-resolution imagery (e.g., PlanetScope or Sentinel-2). The ocean background is complex and dynamic - waves, sun glint and weather effects create clutter that makes reliable detection difficult [2]. Indeed, sea-surface clutter can obscure or mimic the signature of small vessels, leading to false alarms or missed detections under high sea states [2]. These factors, combined with the need to survey vast areas, make automated ship detection in satellite imagery an inherently challenging task.

Conventional deep learning approaches to this problem require large annotated datasets to achieve high accuracy. Labeling satellite images for small objects like ships is labor-intensive and costly, as it demands expert security of high-resolution data and often auxiliary information (e.g. AIS signals) to confirm vessel presence. As a result, obtaining sufficient labeled training data is a major bottleneck [3]. Furthermore, supervised models trained on one region or sensor may not generalize well to new conditions without extensive retraining, due to variations in imaging sensors, environments and acquisition conditions. There is a pressing need for methods that can reduce the dependence on labeled data while improving robustness to varying scenarios.

Foundation models have emerged as a promising solution to these issues. A foundation model is broadly defined as "*any model trained on broad data (typically using self-supervision at scale) that can be adapted (e.g., fine-tuned) to a wide range of downstream tasks*" [4]. In essence, these are large-scale models (often based on CNN or Transformer architectures) pretrained on vast unlabeled datasets, learning rich and generalizable feature representations. They can then be **fine-tuned** with relatively few labeled examples for specific tasks, leveraging the learned general features to achieve strong performance even in data-sparse regimes [3, 5]. Such models have demonstrated remarkable generalization and few-shot learning capabilities in computer vision and natural language domains [3], and recent studies indicate they can set new benchmarks in remote sensing tasks like scene classification, semantic segmentation and object detection [5]. For instance, a model pretrained on hundreds of thousands of unlabeled satellite images can learn invariant features for recognizing ships under

diverse conditions, which a smaller supervised model trained from scratch might not capture.

This work proposes to exploit foundation models for **autonomous ship and maritime object detection** in optical satellite imagery using PlanetScope images. We adopt a two-stage approach: first , an unsupervised pretrained of a large vision model on abundant unannotated satellite (primarily PlanetScope data, with potential extensions to Sentinel-2 or SAR imagery), and second, a supervised fine-tuning of this model on labeled examples of maritime targets. The goal is to transfer general learned features to the downstream task of ship detection, improving detection of small vessels across variable environments while using only a modest amount of labeled data for training. This approach is developed as part of a research and development project at Indra Deimos - **MARVISION**. The MARVISION project has been awarded by the **Spanish Center for Technological Development and Innovation (CDTI)**, with the Ministry of Defense as the end user through the **General Directorate of Armament and Material (DGAM/PLATIN)**, and funded with European funds from the **Recovery and Resilience Mechanism (MRR)** "Next Generation EU". The objective of this contract is the development of a prototype operational system for maritime surveillance and its validation by the Ministry of Defense. This project contributes to the **Aerospace PERTE** in its **Action ACT9 – Spanish Earth Observation System for Security and Defense**, specifically addressing Technological Challenge No. 2. **Indra Deimos**, a Spanish company founded in 2001, employs approximately 500 people across its subsidiaries in 5 European countries. The company operates within the sectors of space, defense, security, transportation, and information technologies. Indra Deimos is a recognized player in the space sector, actively involved across multiple domains, from space exploration and satellite navigation to launchers and space surveillance. However, its greatest expertise is notably in Earth Observation, including the development of satellites and their ground segments, as well as satellite-based applications.

The project is integrated into a full processing pipeline - from raw image acquisition and preprocessing, through detection and identification of ships, to the generation of outputs or alerts for end-users. By training a **foundation model** on broad satellite imagery and specializing it for the maritime domain, we aim to advance the state of the art in satellite-based ship detection, achieving higher accuracy and generalization than conventional methods.

1.2 Objectives

This project explores the feasibility of training a Foundation Model in an unsupervised manner as an initial step to later develop specific detection functions using the learned latent space. The objective is to improve the detection model's learning efficiency, reducing the need for large amounts of labeled data.

Once trained, the model will be integrated into a fully operational processing pipeline, covering the entire workflow from image acquisition to object detection and result delivery.

The specific objectives of this project are:

- **Data Preparation and Management:** Preprocess satellite imagery (PlanetScope mainly) for ingestion into the foundation model and manually annotate

additional samples when required, contributing to the labeling protocol used for supervised learning.

- **Unsupervised Training of the Foundation Model:** Train the model on unlabeled data to learn generalized visual representations and evaluate the quality if the learned features using intrinsic metrics and exploratory analysis.
- **Supervised Fine-Tuning for Detection Tasks:** Fine-tune the pretrained model to detect vessels using a small set of labeled images and evaluate the model performance quantitatively and qualitatively using standard object detection metrics (e.g., precision, recall, confusion matrix, IoU).

This objective are aligned with the overall vision of building a flexible, data-efficient maritime monitoring system leveraging the capabilities of foundation models.

1.3 Report structure

This document is structured as follows:

- chapter 1 **Introduction:** Presents the background and motivation for the project, defines its objectives and outlines the report structure.
- chapter 2 **Related Work:** Summarizes the evolution of ship detection methods in satellite imagery, with emphasis on the recent and state-of-the-art of foundation models and their applications in remote sensing.
- chapter 3 **Materials and Methods:** Describes the data preparation, unsupervised training procedure, model architecture and fine-tuning strategy used to adapt the foundation model for maritime object detection.
- chapter 4 **Experimental Setup and Results:** Details the dataset used, experimental configurations, evaluation metrics and performance results, followed by an analysis of the model's behavior.
- chapter 5 **Discussion and Evaluation:** Discusses the main findings, limitations and implications of the work, with attention to model generalization, data efficiency and operational integration.
- chapter 6 **Conclusions and Future Work:** Concludes the report with a summary of contributions and outlines directions for future research and system development.

Chapter 2

Related work

Maritime vessel detection using satellite imagery is a critical component of modern maritime surveillance. It enables monitoring of vast ocean areas to safeguard shipping routes and detect illicit activities such as illegal fishing, smuggling or piracy [6]. Traditionally, surveillance relied on human operators or shipborne radar, but these methods are labor-intensive and limited in coverage. Advances in Earth observation now provide frequent high-resolution images (optical and synthetic aperture radar) that can be analyzed automatically. However, developing autonomous vessel detection systems remains challenging. Vessels are often small objects in large scene, with high variability in appearance (different sizes, shapes and sensor signatures) and they often appear against complex backgrounds (glinting waves, coastlines) [7]. Environmental variability - such as changing lightning, sea state, weather and sensor conditions - can significantly affect detection performance. For example, waves or sea clutter can cause false alarms and clouds or darkness can obscure optical imagery. Another challenge is data requirements: high performing deep learning models typically need extensive annotated datasets, yet labeling satellite images (marking each vessel with a bounding box or mask) is time-consuming and expensive [7]. This leads to limited publicly labeled data for training, especially for rare classes or new sensors. Computational efficiency is also a concern, as processing large satellite images or continuous image streams in near real-time requires optimized algorithms and hardware. An ideal vessel detection system must balance accuracy and speed, processing gigapixel-scale imagery quickly while minimizing false detections.

In this context, **Foundation Models** have emerged as a promising solution to mitigate some challenges. Foundation models are large-scale neural networks pre-trained on broad data (often through self-supervised or unsupervised learning) that can be adapted to many tasks [8]. In computer vision, this includes models like large Convolutional Neural Networks (CNNs) or Vision Transformers pre-trained on millions of images. Their relevance to vessel detection lies in reducing dependency on labeled data: by learning general visual features from vast unlabeled image corpora, foundation models provide rich representations that can be fine-tuned for ship detection with relatively few labeled examples [8]. Techniques such as contrastive learning and masked image modeling have been shown to significantly improve model robustness and performance on remote sensing tasks (including object detection) by leveraging unlabeled data [8]. These pre-trained representations capture high-level features (like object shape or texture) that help discriminative vessels from background even under varied conditions. In essence, foundation models offer a form of *unsupervised knowledge transfer*, which

can improve generalization to unseen environments and expedite the development of accurate vessel detectors. The following sections provide an overview of the state-of-the-art in satellite-based vessel detection, highlighting existing approaches, the role of foundation models and the comparative merits of different techniques.

2.1 Existing Approaches

2.1.1 Traditional Computer Vision Methods

Early and classical approaches for ship detection in satellite images relied on hand-crafted features and statistical modeling of the imagery. In synthetic aperture radar (SAR) images, a common method is the Constant False Alarm Rate (**CFAR**) detector [9]. CFAR techniques statistically model the sea clutter background and set an adaptive threshold so that only bright targets (e.g., ships) standing out from the background are flagged. Variants of CFAR (such as cell-averaging CFAR or more advanced distributions like Gamma models) were widely used in operational SAT systems due to their simplicity and speed [10]. Similarly, in optical imagery, classical methods included thresholding or edge detection after isolating the ocean region (for instance, using color indices to separate water), followed by morphological operations to identify ship-like blobs. Other approaches used hand-crafted features (e.g., Histogram of Oriented Gradients or texture descriptors) combined with classifiers to detect ships in sliding-window fashion. These traditional methods are efficient and require no training data, but they often struggle with complex scenarios. Sea clutter, sunlight or coastal structures can trigger false detections and small or low-contrast vessels are easily missed. Moreover, fixed decision rules lack adaptability - a CFAR tuned for one sea state may fail in another. Studies have noted that conventional CFAR detectors can suffer from **model mismatch** and low detection rates for small ships in rough conditions [9]. In summary, while classical approaches laid the groundwork for automated ship spotting, their accuracy in diverse environments is limited, motivating the shift toward learning-based methods.

2.1.2 Deep Learning-Based Detection

In the past decade, deep learning has revolutionized object detection in remote sensing, including vessel detection. Modern state-of-the-art methods predominantly use deep neural networks trained on labeled examples to localize ships in images. Two major paradigms are common: **two-stage detectors** and **one-stage detectors** [7]. Two stage detectors as Faster R-CNN first generate region proposals and then classify them as ship or background. These models tend to be very accurate due to their refined, two-step nature, but can be slower in processing large images. One-stage detectors such as Single Shot Detector (SSD) and **YOLO (You Only Look Once)** treat detection as a direct regression problem, predicting bounding boxes and class probabilities in a single network pass [7]. They are generally faster (real-time in some cases) and have become popular for practical maritime surveillance where speed is important. Numerous deep architectures have been explored for vessel detection. For example, the YOLO family (v3, v4, v5, etc.) has been applied to satellite imagery with great success [7]. One study applied YOLOv3, YOLOv4 and YOLOv5 to Airbus's large satellite ship dataset and a Shipsnet dataset, achieving high precision (∼94.7% on a SAR ship test set) after

incorporating attention mechanisms [11]. Another work designed a custom YOLOv7-based model with spatial pyramid clustering and shuffle attention to better detect multiscale ships in complex scenes, and reported improved accuracy in high-resolution ship image datasets (HRSID) [12]. CNN-based detectors have also been combined with tracking algorithms (e.g., DeepSORT) to improve detection consistency in time-series imagery [13]. Beyond CNNs, transformer-based detectors are emerging: for instance, the DETR model (Detection Transformer) pre-trained on large datasets was adapted for ship detection on the SeaShips dataset [14], showing that transformers can serve as powerful backbones for maritime object detection as well.

Several enhancements have been introduced to tailor deep detector to the maritime domain. *Multi-scale feature fusion* (e.g., Feature Pyramid Networks) is commonly used to detect both small boats and large vessel in the same image [15, 16, 17, 18]. *Attention mechanisms* (spatial or channel attention) help the network focus on ship targets against complicated ocean textures [19, 20]. *Data augmentation* (varying brightness, adding noise, flipping, etc.) is employed to simulate different imaging conditions and make models more robust. Despite their high accuracy, deep learning models have known limitations. They are **data-hungry** - training a high-capacity model requires a sufficiently large and representative training set of labeled ship images. If the training data lacks diversity (for example, only ports but no open ocean, or only calm seas), the model may not generalize well to new scenarios [9]. Deep models also tend to treat each pixel grid independently, so they can be fooled by look-alikes (e.g., a cluster of waves or clouds might resemble a ship). Researchers address these issues with techniques like *hard negative mining* (explicitly training on false alarm cases) [21, 22] or *domain adaptation* (adapting a model trained on one distribution, say simulated data or one sensor type, to another.) Indeed, unsupervised domain-adaptive training has been proposed to handle distributions shifts between different satellite sensors or imaging conditions [23]. Overall, deep learning currently offers the best raw detection performance and has become the de facto approach in most recent vessel detection studies.

2.1.3 Hybrid & Specialized Techniques

Some approaches combine classical and deep methods to leverage the advantages of both. For example, a CFAR algorithm might first rapidly scan a SAR image to propose candidate ship pixels, and then a CNN refines those candidates to confirm actual vessels (reducing false alarms) [9, 24]. This can dramatically cut computation by avoiding a full image deep scan while still improving accuracy. Other hybrid strategies include integrating auxiliary data - notably Automatic Identification System (**AIS**) signals - with image detection. Operational systems often correlate satellite detectors with AIS transponder data to validate known vessels and flag "dark" (AIS-silent) targets [25]. In research, AIS data (which provides vessel locations) has been used as a form of weak label to train detectors without manual annotation: if an AIS signal indicates a ship at a location the image patch is assumed to contain a vessel for training purposes. This weakly supervised approach can generate large training sets, though it may introduce noise (e.g., AIS might be present but the ship is visually obscured).

Another line of work is fully **unsupervised vessel detection** - treating ship spotting as an anomaly detection problem. Here, one might model the normal background (open water) and detect vessels as outliers without any labeled examples. A

recent method along these lines is the *Energy Density-Induced Clustering (EDIC)* algorithm [26]. EDIC extracts features via singular value decomposition of local image patches and finds that ship targets have distinct single energy concentrations compared to clutter. By applying K-means clustering on these features (with no ground-truth labels), EDIC can separate ships from sea with high accuracy. Impressively, it was reported to **match or surpass state-of-the-art deep learning detectors** in detection rate and processing efficiency on various SAR scenes [26]. This suggests that carefully designed unsupervised techniques can still compete, especially when labeled data is scarce or when real-time performance on limited hardware is required. Similarly, generative models and autoencoders have been explored to detect vessels as anomalies: for instance, training a deep autoencoder on "empty ocean" images such that when a ship is present, reconstruction error is high, flagging the location. While not yet as prevalent as supervised CNN methods, these unsupervised and hybrid approaches highlight the ongoing innovation to reduce false alarms and dependence on labels.

2.1.4 Foundation Models and Transfer Learning

The concept of foundation models overlaps with the practice of transfer learning that is already common in computer vision, Almost all modern vessel detection networks leverage pre-trained backbones - for example, a ResNet or EfficientNet trained on ImageNet is sued as the feature extractor and then fine-tuned on a ship dataset. This **transfer learning** has been shown to significantly improve performance and convergence speed [21, 22]. In Li *et al.*'s work, they dine-tuined a Faster R-CNN on the SSDD SAR ship dateset with an ImageNet-pretrained CNN and other optimizations, achieving an ∼8% mAP boost over training from scratch [21, 22]. Such results exemplify the value of using a rich foundation of learned features.

Recently, larger and more comprehensive foundation models have emerged in the remote sensing domain [8]. These include models explicitly trained on satellite imagery in an *unsupervised* manner to learn general representations. For example, the **Seasonal Contrast (SeCo)** model (2021) used contrastive self-supervised learning on one million Sentinel-2 images to create a robust ResNet backbone [27]. Likewise, **SimCLR, MoCo, DINO and MAE** frameworks have been applied to build generic feature extractors for overhead imagery [8, 28, 29, 30, 31]. When fine-tuned for tasks like ship detection, such pre-trained models often outperform those trained on natural images because they have already seen a variety of Earth observation conditions (e.g., different water textures, cloud appearances, sensor noise) during pre-training. In one survey, researchers found that self-supervised pre-training (contrastive learning, masked autoencoders, etc.) *significantly enhances object detection performance* in remote sensing by improving model robustness [8, 28, 32, 30, 31]. In practice, this means fewer target-specific labels are needed to reach high accuracy, and the model is more generalizable to new locations or imaging conditions. Foundation models can also be multi-modal: for instance, vision-language models akin to CLIP have been trained on satellite images paired with geo-text (like ship reports or context descriptions) in an unsupervised way [33, 34]. Such models could enable zero-shot detection or classification of vessels by leveraging textual prompts (e.g., detecting "fishing boats" vs "cargo ships" without explicit training on those classes). Another example is Meta AI's Segment Anything Model (SAM) [35], a foundation model for segmentation which, if applied to maritime images, could allow segmenting ships with minimal user input. While these multi-

modal and extremely large models are still cutting-edge research, they represent the next step in applying foundation model advances to maritime surveillance. In summary, existing approaches range from simple thresholding techniques to complex deep learning and foundation model strategies, with a clear trend toward leveraging pre-trained knowledge and unsupervised learning to improve vessel detection performance.

The evolution of vessel detection techniques reflects a trade-off between efficiency, accuracy, and adaptability. Traditional methods, such as CFAR-based detectors, are computationally efficient but struggle with false alarms in cluttered maritime environments. Deep learning approaches, including YOLO and Faster R-CNN, significantly enhance detection accuracy but require large-scale labeled datasets and high computational resources. In contrast, foundation models leverage self-supervised learning to extract rich representations from unlabeled satellite data, enabling few-shot learning and better generalization across different imaging conditions. However, their deployment is constrained by high pre-training costs and large memory footprints. Table 2.1 summarizes the key strengths and limitations of these approaches, highlighting their respective roles in modern vessel detection systems.

Approach	**Strengths**	**Limitations**
Traditional Methods (CFAR, thresholding)	Fast, no training data required	Poor accuracy in cluttered scenes, high false alarms
Deep Learning (YOLO, Faster R-CNN)	High precision, robust to noise	Requires labeled data, computationally expensive
Foundation Models (Self-supervised, contrastive learning)	Fewer labeled samples needed, strong generalization	High pre-training cost, large memory footprint

Table 2.1: Summary of strengths an limitations of current existing approaches.

2.2 Comparative Analysis

To understand the state-of-the-art, it is useful to compare key detection techniques across several criteria: **detection accuracy, computational efficiency, generalization, adaptability** and **scalability/deployment**. Table 2.2 provides a high-level comparison of representative approaches. The discussion of these aspects are here below:

- **Detection Accuracy:** Modern deep learning models offer the highest accuracy for vessel detection. One-stage detectors like YOLOv5 and two-stage detectors like Faster R-CNN can achieve high precision and recall on benchmark datasets (often exceeding 90% precision/recall in favorable conditions) [7, 36, 11, 37, 38, 39]. For example, a tailored YOLOv5 variant reached 94.7% precision on a standard SAR ship dataset [36]. By contrast, traditional methods have moderate

accuracy. A basic CFAR detector might miss dim or small vessels and is prone to false alarms from sea clutter [9]. Unsupervised clustering methods, while improving, typically lag supervised CNNs in detection rates *unless* carefully engineered (EDIC being a notable exception that reported comparable accuracy to deep models) [26]. **Foundation models** enhance accuracy especially when labeled data is limited: a fine-tuned foundation model can maintain high recall where a standard model might fail, because its pre-trained features better distinguish ships from novel backgrounds [8]. In summary, supervised deep models currently set the state-of-the-art accuracy, with foundation model pretraining boosting their performance further, whereas classical detectors remain inferior in complex scenarios.

- **Computational Efficiency:** Simpler methods have the edge in speed. CFAR and another thresholding techniques run in near real-time even on large images (they involve few operations per pixel) and can be deployed on low-power devices. Deep learning models are more computationally demanding, often requiring GPU acceleration dor real-time inference. Among deep models. one-stage architectures (YOLO, SSD) are optimized for speed - e.g., YOLOv4 can process ∼65 frames per second on COCO-sized images [7, 40, 41], and even lightweight versions (YOLOv4-tiy) reach hundreds of FPS with some accuracy sacrifice [7, 42]. Two-stage detectors are usually slower (Faster R-CNN might run a few FPS or less, due to the proposal stage) [7]. The largest foundation models (e.g., Vision Transformers with billions of parameters) incur additional computational cost during inference [14], but in practice many foundation models used for detection (like ResNet50 with self-supervised weights) have inference cost similar to a normal CNN. Therefor, using a foundation model does not drastically change runtime; the heavy cost is in the *pre-training* phase which is done offline. Recently research also explores model compression and efficient backbones (MobileNets, knowledge distillation) to deploy vessel detectors on edge devices (drones, nanosatellites) [43, 44, 45]. A point worth noting is that some unsupervised methods are extremely fast: the EDIC clustering method, for instance, involves a singular value decomposition on local windows and k-means clustering, which is much lighter than a deep CNN forward pass, thus offering high throughput [26]. In summary, classical detectors are most efficient, one-stage deep detectors strike a good balance of speed and accuracy and foundation-based detectors have similar runtime as their architecture type (with potentially larger memory footprint).

- **Generalization to Unseen Data:** Generalization refers to how well a method performs on data that differ from its training or design conditions (e.g., a new geography, sensor or weather situation). Here, *foundation models and unsupervised approaches offer a clear advantage.* Because foundation models are pre-trained on diverse dataset (potentially including multiple geographies and seasons), they imbue the detector with a broad prior knowledge. Fine-tuned foundation models have demonstrated greater robustness to distribution shifts than models trained from scratch [8]. For example, a model pre-trained via contrastive learning on global satellite imagery can detect ships in an Arctic scene or a new sensor's images more reliably than an purely supervised model that has never seen such cases. Traditional methods based on physical models (like CFAR) are *implicitly general* in that they don't learn from data at all - a CFAR detector will apply

the same statistical threshold logic anywhere. This can be a strength (no risk of overfitting to a dataset), but in practice these methods need manual parameter tweaking for each new condition (sea clutter statistics vary, etc.), so they are not automatically adaptive. Supervised deep learning detectors have *notorious generalization issues* when applied to unseen domains: a network trained on one satellite or region often sees performance drop on another due to differences in image resolution, noise or vessel appearance [9]. This key concern in maritime monitoring, since one may need the model to work globally. Techniques like domain adaptation and augmentation help, but they require effort. **Adaptability** is related - how easily a method can be adapted or fine-tuned with minimal labeled samples. A few-shot fine-tuning (or even zero-shot with the right prompt, in the case of vision-language models) can quickly adapt a foundation model to a new scenario (e.g., fine-tuning on just a handful of SAR images to detect ships in SAR after being pre-trained on optical images). Standard deep models can also be adapted, but with more labeled data usually. Classical methods are the least adaptable in an automated sense; adapting them means manually recalibrating parameters.

- **Scalability and deployment Feasibility:** Scalability considers both the ease of deploying the model in real operational systems and how performance scales with large data. Classical methods are extremely easy to deploy - they run on basic hardware and have transparent mechanisms. They scale linearly with image size and number of images and can be parallelized over large image archives without intensive computation infrastructure. This is why even today, simple threshold-based detectors are sometimes used for initial scan of massive satellite archives. Deep learning methods require more complex deployment: a GPU server or cloud service is typically needed to process images in bulk, especially if using high-resolution imagery. However, modern object detectors can be packaged in lightweight forms (e.g., as ONNX models) [46] and accelerated by AI chips, making deployment on satellites or aircraft increasingly feasible. For instance, companies have demonstrated on-board ship detection on micro-satellites using compact neural networks [47]. **Real-time alerting** is achievable with one-stage models given sufficient hardware. Foundation models, if extremely large (like a very deep transformer), might be impractical to run on the edge, but one can often use a *distilled* or smaller version for deployment. Moreover, foundation models can be shared as a common resource (e.g., a cloud API providing a pre-trained model that can be fine-tuned for a client's data). This centralizes the heavy computation and lets many users benefit from the same core model, In terms of *scaling to many targets or wide areas*, deep learning models have an advantage of maintaining high accuracy even as scene complexity grows (they can handle many instances in one image). Traditional methods might trigger many false alarms in busy scenes (e.g., a harbor with dozens of closely spaced vessels). In such scenarios, learning-based model that use context and learned features handle dense targets better.

Overall, the comparative landscape shows that **deep learning with transfer learning** currently offers the best accuracy, while **classical methods** win in simplicity and speed. **Foundation model approaches** strive to combine the best of both - high accuracy and generalization, with less training data burden - at the cost of

initial complexity and resources for pre-training. As computing hardware continues to improve and large models become more accessible, it is expected that foundation-model-powered detectors will be increasingly favored for their superior robustness in real-world maritime surveillance.

Approach	Detection Accuracy	Computational Efficiency	Generalization & Adaptability	Scalability & Deployment
Traditional (Rule-based) *e.g., CFAR thresholding*	Moderate in simple scenarios; struggles with small or low-contrast vessels. High false-alarm risk in complex backgrounds [9].	**Very high efficiency:** Fast runtime, minimal resource needs (can run on CPU). Suitable for real-time scanning of large areas.	Relies on fixed statistical models - not data-driven. Requires manual retuning for different sea states or sensors (limited adaptability).	Easy to deploy on any platform (even on-board satellites) due to low resource usage. Scales linearly with image size; but may produce many false detections in crowded scenes.
Deep Learning (Supervised) *YOLO, Faster R-CNN*	**State-of-the-art accuracy:** High precision and recall (often ¿90% on benchmarks) [7]. Detects small and large vessels with learned features, vastly outperforms classical methods in challenging conditions.	**Moderate efficiency:** Requires GPU or neural accelerator for real-time performance. One-stage models (YOLO) achieve near real-time inference [7]; two-stage models are slower. Computational cost grows with size and model complexity.	Good generalization *if* training data covers the scenario. Otherwise, performance can drop on unseen conditions [9]. Adaptable through retraining or fine-tuning, but needs labeled data. Some robustness gained via data augmentation and regularization.	Deployment is feasible on servers or edge devices with AI chips. Models must be quantized/optimized for edge use. Scalability depends on computing infrastructure; processing large image volumes is costly but parallelizable. Widely adopted in operational systems with cloud support.
Foundation Model-Based *e.g., Pre-trained/self-supervised backbone*	**High accuracy even with limited new labels:** Pre-training on diverse data yields robust features, improving detection of difficult targets (fewer missed vessels) [8]. Approaches supervised baseline performance, sometimes exceeding it in low-data regimes.	**High training cost**, but **inference cost varies:** Pre-training (unsupervised) is computationally intensive (done offline). Inference can be on par with standard deep models if using a similar architecture (e.g., a ResNet50 backbone). Very large models may be slower, but techniques like model distillation can alleviate this.	**Strong generalization:** Broad pre-training confers resistance to distribution shifts (sensor, location) [8]. Readily adaptable via fine-tuning or even zero-shot/prompt-based methods. Can quickly learn new vessel types or conditions with minimal additional data.	Requires access to pre-trained model (often large). Deployment of a huge model on the edge is challenging, but feasible via cloud or by using compressed versions. Scales well in the sense that one foundation model can serve many tasks/users. As data grows, the same model can be continually refined rather than training separate models for each new dataset.

Table 2.2: Comparison of vessel detection approaches.

2.3 Evaluation Frameworks

2.3.1 Datasets

A variety of datasets and benchmarks have been established for training and evaluating vessel detection algorithms. These include both synthetic aperture radar datasets and optical imagery datasets, each with its own characteristics:

- *Synthetic Aperture Radar (SAR) Datasets:* The **SAR Ship Detection Dataset (SSDD)** is a prominent benchmark in this category. SSDD, first released in 2017 and is considered the first large open dataset for SAR ship detection and has been used by nearly half of published studies in this area [21, 22]. It contains SAR images with annotated ship bounding boxes. An official improved release of SSDD provides standardized training/test splits and annotations (including oriented boxes and segmentation masks) to enable fair comparisons [21, 22]. The popularity of SSDD has spurred the creation of other SAR datasets: **HRSID** (High-Resolution SAR Images Dataset) includes 5,604 SAR image patches and over 16,000 ship instances, focusing on high-resolution and even small ships [48]. **LS-SSDD-v1.0** [49] is a large-scale expansion of SSDD with Sentinel-1 imagery targeting small vessels in big scenes. Additionally, datasets like AIR-SARShip (versions 1.0 and 2.0) [50, 51] provide SAR imagery with ships, some focusing on airborne SAR. These SAR datasets allow algorithms to be benchmark under different conditions (e.g., varying resolutions, clutter environments). They typically provide ground truth in the form of bounding boxes (horizontal or rotated). Evaluation on SAR benchmarks often considers *false alarm rate* in addition to precision/recall, since an important metric for radar surveillance is how many false detections occur per image or per area, given a detection threshold.

- *Optical Satellite Datasets:* In the optical domain, early datasets were relatively small. **ShipsNet** (from Kaggle's "Ships in Satellite Imagery") [52] is one example: it contains 4,000 80x80 RGB images labeled with either a "ship" or "no-ship" classification. A far larger contribution came from the **Airbus Ship Detection Challenge (2018)** [53], which released a dataset of ∼40,000 satellite images (from Airbus SPOT 6/7 at 1.5m resolution) with segmentation masks for ships. This dataset (often just called the "Airbus ship dataset") has been widely used to develop and evaluate segmentation-based detectors [7, 11]. Following this, researchers built more specialized datasets. **MASATI (Maritime Satellite Imagery)** [54, 55] is a dataset of 7,389 satellite images labeled according to the following seven classes: land, coast, sea, ship, multi, coast-ship, and detail. **HRSC2016** and **VHRShips** are high-resolution aerial/satellite image datasets for ship detection/classification [56] - they include many ship types and oriented bounding box annotations, though they often include harbor scenes which are easier (ships large and well-separated from background). A recent dataset called **ShipRSImageNet** [57] compiles 3,435 satellite images from various sensors and locations, with around 17,000 ship instances in 50 categories, aiming for diversity (it's been touted as one of the largest ship detection datasets before 2024).

- *Integrated and New Datasets:* The field has seen a trend toward larger and more comprehensive benchmarks. Notably, **UOW-Vessel** [56] is a benchmark of high-resolution optical satellite images explicitly for vessel **detection and segmentation**. It consists of 3,500 large images collected from 14 countries, with 35,598 vessel instances in 10 categories (e.g., cargo ship, tanker, fishing boat, etc.). Uniquely UOW-Vessel provides precise polygon annotations for each ship (rather than just bounding boxes), enabling evaluation of instance segmentation - a more fine-grained detection task relevant for identifying ship outlines [56]. This dataset is currently the largest of its kind in the optical domain and allows researchers to test algorithms on a variety of conditions worldwide. Apart from standalone

datasets, some challenges like **Xview** [58] and **DOTA** [59] include "ship" as one of multiple object classes in overhead imagery. For example, DOTA (Dataset for Object Detection in Aerial Images) has thousands of images with objects ranging from vehicles to buildings and ships, providing a broader test of detection algorithms in mixed contexts [59].

Each dataset typically comes with a defined train/test split and sometimes a *benchmark leaderboard*. Using common datasets allows fair comparison of algorithms. It's important to note dataset biases: e.g., methods tuned on the calm, largely clear images of the Airbus dataset might falter on the more challenging MASATI images with tough seas and clouds. Therefore, robust evaluation should test an algorithm on multiple datasets or on a held-out region it hasn't seen in training.

2.3.2 Evaluation Metrics

To quantitatively asses detection performance, standard metric from object detection and segmentation are used:

- **Precision and Recall:** Precision (the fraction of detection that are correct) and recall (the fraction of true vessels that are detected) are fundamental metrics. High precision means few false alarms, and high recall means few missed targets. Often a Precision-Recall curve is plotted, and the balance of the two is considered. In maritime surveillance, precision is crucial to avoid overwhelming operators with false alerts, but recall is equally critical since a missed detection could mean an undetected illicit vessel.

- **F1-Score:** This is the harmonic mean of precision and recall. It provides a single measure that balances the two (F1 is high only if both precision and recall are high). Some papers report F1-score at a particular operating threshold, especially if optimizing for a trade-off.

- **Intersection over Union (IoU):** In object detection, a detection is considered a "true positive" if the overlap between the predicted bounding box and the ground truth box exceeds a threshold (commonly 0.5 or 50%). IoU is the area of overlap divided by area of union of the boxes. This metric is also used for segmentation mask overlap. IoU is important because it measures how accurately the location and size of the vessel is predicted, not just whether the object is found. Competitions often use IoU thresholds (e.g., IoU ≥ 0.5) to define correct detections.

- **Average Precision (AP) and mean AP:** Average Precision is the area under the precision-recall curve for a given class. In detection challenges (e.g., COCO [41] or DOTA [59]), one computes AP for each class and the takes the mean (mAP). For ship detection where there is essentially one class ("ship"), AP is equivalent to the area under the PR curve for that class. Many works report mAP at IoU=0.5 (Pascal VOC criterion [60]) or the COCO-style mAP which averages AP across IoU thresholds from 0.5 to 0.95. For instance, one study reported YOLOv3 achieving 63% mAP and RetinaNet 79% AP on a SAR ship dataset [7, 20], indicating RetinaNet was more accurate in that test. Higher AP/mAP indicates better overall detection performance.

- **False Alarm Rate:** Especially for surveillance applications, false alarm or false positive rate is monitored. This could be given as number of false detections per image or per square kilometer. A low false alarm rate is crucial in practice to ensure trust in the system outputs. Some papers report the probability of detection (Pd) at a certain false alarm rate (Pfa), which comes from radar community metrics.

- **Computational Metrics:** To evaluate efficiency, metrics like *inference time per image, frames per second (FPS),* or *GFLOPs* are cited. For example, a method might be said to run at 10 FPS on 1024x1024 images on an NVIDIA V100 GPU. Model size (number of parameters or memory footprint) is also noted when relevant, as it affects deployability. The UOW-Vessel paper, for instance, provides model size and inference speed for tested algorithms as part of the benchmark to compare not just accuracy but also resource usage [56].

- **Robustness Metrics:** Some evaluations include how performance changes with perturbations (like adding noise, varying resolution) to assess robustness. While not a single metric, it's an evaluation strategy to test generalization.

Ground truth annotation quality heavily influences these metrics. One challenge in vessel datasets is **annotation consistency** - e.g., how to mark a ship that is only partly visible at the image edge, or two ships very close together. Inconsistent or coarse annotations (like a single box covering two adjacent boxes) can penalize an algorithm unfairly or make the metric less informative. The SSDD initial version had some coarse labels that were later refined in the official release for more precise evaluation [21]. Similarly, differences in annotation formats (bounding box vs polygon) mean that algorithms should ideally be evaluated in the same format they output.

2.3.3 Challenges in Annotation & Validation

Labeling satellite images for vessel detection is labor-intensive. Large-scale dataset often use a combination of manual annotation and semi-automated tools. For instance, the Airbus challenge [53] provided segmentation masks that were likely produced by manual tracing and quality control on thousands of images - a huge effort. In some cases, annotation errors exist (e.g., a ship missed by annotators or false label for something that is not a ship). This can impact validation: an algorithm might detect a real ship that the ground truth missed and be penalizes as a "false positive" when it was actually correct. To mitigate this, some evaluation protocols allow a tolerance for unlabeled objects or employ multiple human reviews of test images. Cross-validation (splitting data into train/val folds) is used when dataset size is limited, to ensure results are not a fluke of a particular split.

Another validation strategy is **cross-dataset evaluation:** train in one dataset and test on another to gauge generalization. For example, training on MASATI and testing on Airbus, or vice versa. f a model still performs well, it indicates robustness. This kind of evaluation is increasingly encouraged given the desire for models that work on global scale data.

In summary, rigorous evaluation of vessel detection involves using standardized datasets with clear train/test splits and reporting metrics like precision, recall, F1 and AP/mAP, along with computational efficiency. Proper validation should consider the

ambiguities in ground truth and test the model's ability to handle the variety of real-world conditions. Community benchmarks (leaderboards, challenges) have been key in pushing the state-of-the-art by providing a common reference for comparison.

2.4 Conclusions & Future Directions

2.4.1 Summary of Findings

The state-of-the-art in vessel detection from satellite imagery has advanced significantly, achieving capabilities that were unattainable with older methods. Deep learning detectors now form the backbone of most high-performance systems, demonstrating excellent accuracy in both optical and SAR imagery for vessel localization. Techniques like YOLO, Faster R-CNN and transformer-based models have been tailored to maritime scenarios with specialized modules for small object detection and complex background suppression. These methods dramatically outperform classical techniques (like CFAR or simple image processing), especially in challenging environments (rough seas, cluttered coastlines, low-contrast targets). At the same time, we identified **gaps and limitations** that persist. Chief among them is the dependence on large labeled datasets - a barrier in a domain where labeling is difficult. Many current model struggle to generalize beyond the conditions they were trained on. For instance, a model trained on a calm Atlantic water may stumble when faced with SAR images of the polar seas or an optical image taken under different illumination. False positives remain an issue, as algorithms can still confuse unusual robustness. Moreover, deploying these advanced models in real operational pipelines (with strict real-time and reliability requirements) is non-trivial due to computational demands and the need for validation for results (analysts often require high confidence in alerts).

2.4.2 The Role & Impact of Foundation Models

Foundation models have emerged as promising avenue to address some of these challenges. By leveraging unsupervised learning at scale, they reduce the need for labeled data and provide a strong starting point for many tasks. In the context of maritime vessel detection, foundation models are expected to *significantly impact future systems* in several ways. First, their use will likely **reduce the training data bottleneck** - instead of requiring thousands of annotated images for a new satellite or scenario, a pre-trained foundation model can be fine-tuned with a few dozen examples, making it feasible to develop detectors for niche scenarios (e.g., specific coastal region or a new satellite sensor) quickly and with lower cost. Second, foundation models trained on diverse data can serve as a **universal backbone** for maritime vision tasks. We can envision a single large model that has ingested multi-spectral satellite imagery, aerial imagery, perhaps even ship photographs and AIS text descriptions to build a holistic representations of vessels. Such a model could be adapted not only to detect ships, but also to classify types, segment their outline or even predict their intents, all with minor task-specific tuning. This cross-task adaptability would streamline the development of comprehensive maritime surveillance systems (detection, identification, tracking all using the same core model).

Another advantage is **improved generalization and robustness**. Foundation models, by virtue of being trained on "everything", tend to capture features that

are invariant to many changes (lighting, viewpoint, noise). Future vessel detectors built on foundation models are expected to be more resilient to distribution shifts - for example, a foundation-model-based detector might handle a sudden change in image quality or an unseen environmental condition more gracefully, having effectively "seen" something similar in its vast pre-training experience [8]. This will be crucial for building truly global maritime monitoring capabilities that work 24/7 under all weather. Early evidence of this is seen in research where self-supervised pre-trained led to more robust ship detection detection in stormy weather images than a conventionally trained model [8].

2.4.3 Future Research Directions

Building on current progress, several directions appear promising, One is the development of *multi-model foundation models for maritime surveillance*. This involves combining data streams - for instance, training a model on satellite imagery together with AIS data, ship logs or radar data - so that the model learns the correlation between visual cues and other information. A vision-language foundation model that understands a sentence like "a tanker near port with oil slick" and associates it with satellite imagery could enable powerful query-based surveillance (e.g., ask the model to find "possible oil-smuggling vessels" in imagery). Some recent work has already demonstrated unsupervised vision-language alignment in remote sensing [33], which could be extended to maritime scenarios.

Another direction is *real-time adaptable models*. Rather than training a static model and deploying it, future systems might employ models that continue to learn on the fly in an unsupervised manner. A satellite imaging a new region could use self-supervised learning to adjust its vessel detection model to local background characteristics without explicit labels. This concept of **continuous learning** or life-long learning will leverage foundation model architectures that can be updated with new data streams. It could help address the issue of domain shift continuously (for example, a model that gradually adapts from summer imagery to monsoon-season imagery as the year progresses, without needing a full retraining).

On the algorithmic front, there is interest in bridging the gap between fully supervised detectors and unsupervised anomaly detectors, Semi-supervised learning methods - where a small labeled set and a large unlabeled set are used together - could dramatically increase detection performance when annotated data is scarce. Foundation models provide one way to do this (by pre-training on the unlabeled set), but other techniques like teacher-students self-training ir active learning (where the model identifies uncertain detections for human to label) can be incorporated into future workflows.

2.4.4 Expected Impact on Maritime Surveillance Systems

As foundation model techniques mature, we expect next-generation maritime surveillance systems to be far more **autonomous, scalable and intelligent**. The dependency on human analyst for constant monitoring will lessen as detectors become more reliable with fewer false alarms. Analysts will instead focus on reviewing high-confidence alerts or handling the integration of multi-source data, with AI handling the grunt work of scanning thousands of images. Large-scale monitoring - for example, tracking every vessel above a certain size across all oceans - becomes more feasible

when the detection models are both accurate and efficient (which foundation models can facilitate by enabling use of smaller fine-tuned models without losing accuracy). Additionally, the **scope of detection will broaden**. Instead of just finding whether a vessel is present, systems will also recognize vessel types, activities (e.g., transshipment at sea, formation patterns), and anomalies (a ship that deviates from normal routes or behaves erratically). These higher-level capabilities can be achieved by building on the rich representations of foundation models and training them for tasks beyond just detection.

In conclusion, the state-of-the-art in vessel detection has reached a point where operational use of AI is reality, and the incorporation of foundation models is set to push it even further. By overcoming data limitations and improving adaptability, foundation model-based approaches will likely become the **cornerstone of maritime AI systems**. Researchers will focus on refining these models, addressing any remaining shortcomings (such as interpretability and trustworthiness of the AI decisions), and integrating them seamlessly with existing surveillance infrastructure. The outcome will be more secure and well-monitored seas, where illicit or dangerous activities can be detected early and accurately through the collaborative efforts of advanced AI and human oversight.

Chapter 3

Materials & Methods

3.1 Data & Pre-processing

3.1.1 PlanetScope SuperDove Imagery

This project utilizes PlanetScope SuperDove satellite images, which provide ∼3m per pixel resolution and multispectral data across 8 bands [61]. The standard bands include **Blue, Green, Red and Near-Infrared (NIR)**, and the SuperDove generation adds four additional bands: **Coastal Blue, Green I, Yellow and Red Edge** [61]. We obtained PlanetScope scenes as surface reflectance products, meaning the imagery had been atmospherically corrected to approximate true surface reflectance values. Such correction removes atmospheric effects and ensures radiometric consistency across images [62]. By using Planet's surface reflectance data (e.g. the *AnalyticMS SR* 8-band product), we minimize illumination and atmospheric variability between scenes.

3.1.2 Spectral band Selection

Of the 8 available bands, we excluded the Green I (531nm) and Yellow (610nm) bands from our analysis, reducing the data to 6 spectral channels. The Green I and Yellow bands are unique to PlanetScope and have no direct equivalent in Sentinel-2 or other mainstream sensors, which complicates cross-sensor calibration [62]. Moreover, Planet's documentation notes that six out of the eight SuperDove bands are designed to be interoperable with Sentinel-2 bands [62]. by removing Green I and Yellow, we focus on the core set of six well-calibrated bands (**Coastal Blue, Blue, Green, Red, Red Edge, NIR**) that align with conventional multispectral channels. All images were rechecked after band selection to ensure no data misalignment occurred.

3.1.3 Cloud Masking and NoData Handling

We applied Planet's quality masks (UDM2) to filter out unusable pixels (e.g. clouds or void areas), marking those as NoData. Consequently, each scene's pixels are categorized as either valid data (with reflectance values) or NoData (masked). We used masked array structures in Python to handle these, so that NoData values would not interfere with statistical computations. In particular, NoData pixels were ignored during normalization and excluded from model training. Any pixel with no valid reflectance (or set to fill value 0) is treated as masked in subsequent processing.

3.1.4 Radiometric Normalization

To put multi-scene imagery on a comparable scale and mitigate extreme outliers, we performed per-band normalization using robust percentile stretching. For each spectral band, we computed the 2nd and 98th percentile values across all valid pixels (ignoring zeros/NoData and NaNs). Pixel values below the 2nd percentile were raised to the 2% value, and those above the 98th percentile were lowered to the 98% value, effectively clipping the tails of the distribution. We then linearly scaled each band so that the 2nd percentile maps to 0.0 and the 98th percentile to 1.0. Mathematically, for a given band we calculate `pct2` and `pct98`, and apply:

$$x_{\text{norm}} = \frac{x - \text{pct2}}{\text{pct98} - \text{pct2}} \tag{3.1}$$

clipping the result to [0, 1]. This yields dimensionless normalized reflectance values. By excluding zero values (which indicate NoData) from the percentile calculation, we ensure that masked regions do not skew the statistics. After normalization, NoData pixels remain at 0 (since they were initially 0 and get clipped to 0), whereas valid data are in the [0, 1] range. This normalization strategy (sometimes called percentile clipping) preserves relative reflectance differences while reducing the influence of extreme pixel values (e.g. glare or deep shadows).

3.1.5 Patch Tiling

Given the high resolution and large size of PlanetScope scenes, we adopted a patch-based approach for model training. Each corrected and normalized image was divided into non-overlapping square patches of 128×128 pixels (which at 3m resolution corresponds to roughly 384×384m on the ground). We slid a fixed 128px grid over each scene without overlap, partitioning it into an array of patches. Only full patches of size 128×128 (no partial edges) were considered, effectively discarding a thin border if the image dimensions were not exact multiples of 128. We further filtered patches to retain only those that were entirely valid (10% NoData-free). In practice, for each candidate patch we checked that it contained no masked pixels – if any pixel was flagged as NoData (value 0 after normalization), that patch was excluded from the "full" training set. This was implemented by verifying `np.all(patch > 0)` on the normalized patch array. Patches that passed this criterion were saved for use in model training (others were either discarded or kept only for analysis). The rationale was to avoid introducing any cloudy or missing data during training, so the model learns from clear imagery only in a first training. As a result, our foundation model pretraining used the subset of patches with 100% valid data, ensuring that the network isn't distracted by blank or corrupted inputs. This yielded a large collection of image patches (on the order of tens of thousands of patches in total) for the self-supervised learning stage. Then, the foundation model was retrained with all kind of data to ensure there is noi bias in the first training and to make the model more robust in real-case scenarios.

Patch validity thresholds.

During preprocessing three candidate datasets were produced, keeping only patches with at least 80%, 90%, or 100 % valid pixels, respectively. Although the 80% and 90% sets increased the sample count by 15–25%, early experiments revealed a slight

degradation in performance when mixed valid/NoData patches were included *and validation was carried out on fully valid images*. Evaluating exclusively on ideal validation data can introduce bias, as the model's robustness to partially masked inputs remains untested. Consequently, only the 100%-valid set was retained for the results reported in this thesis, while acknowledging that an additional validation fold containing partially invalid patches would be required for a fully unbiased assessment.

3.1.6 Label Generation

Supervised training (for ship detection) requires ground-truth labels for ships in the imagery. These labels were provided in vector form (GeoJSON polygons delineating each ship's outline or location). We converted the vector annotations into raster masks aligned with our image patches. Using the geospatial metadata (coordinate reference system and affine transform) from the PlanetScope images, we rasterized the ship polygons to binary masks at the same 3m resolution. Each mask pixel is set to 1 if it falls inside a ship polygon and 0 otherwise. This process was done on a per-patch basis: for each 128×128 patch, we took the corresponding subset of ship polygons (those overlapping that patch's area) and burned them into a 128×128 binary mask. All masks and images share the same CRS and alignment, so that a ship's pixels in the image correspond exactly to 1's in the mask. We utilized the Rasterio library's functionality (specifically `rasterio.features.rasterize`) [63] to perform this conversion reliably. CRS transformations (if needed) were handled so that vector and raster data matched coordinate systems as shown in Figure 3.1. The end result is a set of training image patches and matching label masks for supervised learning: each image patch has a corresponding 128×128 array of labels (1 for **vessel** pixel, 0 for **water**). If a patch contained no part of any ship, its mask would be all zeros (we handle such cases in training as described later).

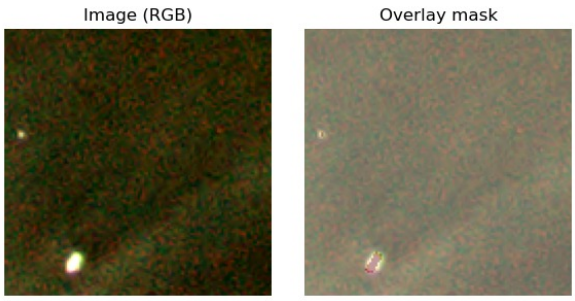


Figure 3.1: PlanetScope raw image with vessels and labels matching marked in red (as they have the same CRS).

3.1.7 Note on Tools - Introduction to Rasterio

All the geospatial preprocessing above was implemented in Python using the Rasterio library [63]. Rasterio is a Python API built on GDAL that provides easy access to geospatial raster data using NumPy arrays. In our workflow, Rasterio was used to read PlanetScope GeoTIFF images into memory as masked numpy arrays (honoring the NoData mask) and to write out processed patches. It also facilitated coordinate transformations and rasterization of vector labels. For example, reading an image with

`rasterio.open()` yields the pixel array along with metadata like transform (mapping pixel indices to real-world coordinates) and CRS. We leveraged these to ensure that when we rasterized ship polygons, we used the correct transform and shape matching the image patch, producing an accurate label mask. In summary, Rasterio allowed us to treat satellite imagery as just arrays (for normalization, tiling, etc.) while preserving the georeferencing under the hood – a vital capability when aligning ground-truth vectors with imagery. This brief introduction is provided as many deep learning practitioners may be unfamiliar with satellite-specific libraries; in essence, Rasterio simplifies remote sensing data handling by exposing it in a scientist-friendly manner (NumPy arrays and GeoJSON interoperable) [63].

3.2 Model Architecture & Self-Supervised Pretraining

3.2.1 U-Net Architecture (Residual Encoder-Decoder)

The architecture of the segmentation model is based on the well-known U-Net convolutional network. In particular, we employ an **enhanced version of the U-Net model used in the PhilEO benchmark** [64], which retains the fundamental encoder–decoder structure characteristic of U-Net. This means the network has a contracting path that captures multi-scale features and an expanding path that enables precise spatial reconstruction of the segmentation map. U-Net architectures are a popular choice for semantic segmentation in Earth observation tasks [64], and by building on this foundation we ensure the model benefits from proven design principles while adapting it to our specific needs.

Several **structural adaptations** have been introduced to the baseline U-Net with the goal of **improving feature extraction capacity** in the context of large-scale pretraining and the target task of ship segmentation. These architectural modifications (omitted here for confidentiality) strengthen the network's ability to learn rich and discriminative representations from satellite imagery. In practice, the improved design allows the model to better capture subtle details of ships and robustly integrate contextual information across different scales, which is crucial for accurate segmentation. By leveraging the strengths of the original U-Net and carefully refining its components, the resulting model is better tailored to the maritime domain and pretraining scenario, ultimately achieving superior performance on ship segmentation tasks compared to the standard U-Net baseline [64].

Figure 3.2 provides a generic schematic of the original U-Net architecture introduced by Ronneberger et al. (2015) [1]. Although our implementation incorporates proprietary adaptations, the high-level encoder–decoder structure with symmetric skip connections remains conceptually similar.

3.2.2 Self-Supervised Pretraining Procedure

During the self-supervised pretraining stage, we trained the U-Net model on a large set of unlabeled PlanetScope image patches, using a masked reconstruction objective as described above. To improve the model's robustness and encourage it to learn generalizable features, we applied data augmentation to the input patches in this phase. In

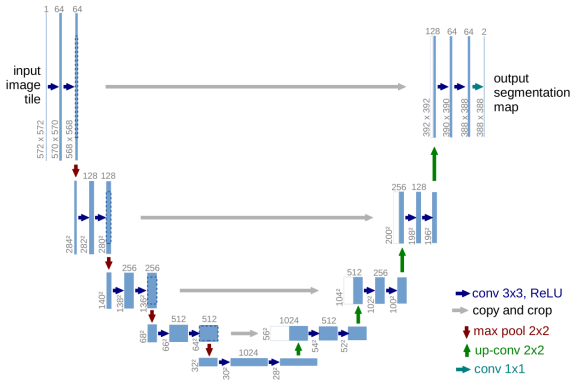


Figure 3.2: Generic U-Net encoder-decoder architecture. The contracting path (left) successively downsamples the input tile by max-pooling, while the expanding path (right) upsamples and concatenates feature maps via skip connections to reconstruct a full–resolution segmentation mask [1].

particular, we used common geometric augmentations such as random rotations (e.g., 90° increments or small arbitrary angles) and horizontal/vertical flips, a standard regularization technique in computer vision [65]. Each training patch had a random chance of being rotated or flipped before masking and feeding it to the network, effectively increasing the diversity of views seen during pretraining. These augmentations help the encoder learn invariant feature representations (for example, a ship looks like a ship regardless of orientation), which is beneficial for downstream detection. We note that no color or brightness augmentations were applied (since the focus was on geometric invariances), and that all augmentations were applied only during the foundation model training.

Training of the foundation model was carried out using a reconstruction loss computed only on the masked-out pixels (the model's task being to predict those missing pixel values). We employed a mean squared error (MSE) loss between the decoder output and the ground truth pixels in the masked regions. The model was optimized using the Adam optimizer, and we trained for several hundred epochs until convergence (details of hyperparameters are provided in Section 3.5.2). Through this unsupervised training, the encoder gradually learned to capture essential scene content in its latent features in order to fill in missing areas – effectively learning a rich representation of PlanetScope imagery without any manual annotations.

*(**Note:** Details of the network architecture – number of layers, feature dimensions, etc. – are kept at a high level for confidentiality. The key point is the encoder-decoder design and the masked reconstruction learning scheme, which are general techniques in current deep learning research.)*

3.3 Fine-Tuning for Ship Detection

After pretraining, the model was adapted and fine-tuned for the specific task of ship segmentation in PlanetScope images. We leveraged the same U-Net architecture described in Section 3.2.1, using the pretrained encoder as the backbone for feature extraction. The decoder for this segmentation phase was newly initialized (while sharing the same

structural design of convolutional, residual, and attention blocks as the pretraining decoder) and produces a binary segmentation mask as output. In essence, the fine-tuning model is architecturally identical to the pretraining model, except that its final output layer is a 1×1 convolution with sigmoid activation yielding a probability map (where each pixel indicates the confidence of being vessel or background). The encoder weights were initialized from the foundation model, providing a strong starting point for the supervised learning. During fine-tuning training, we did not apply any data augmentations to the input patches. This is a deliberate choice to ensure that the supervised training data (which are relatively limited and precisely annotated) are used in their original form. Each training iteration involved feeding a batch of real, 128×128 patches (each containing at least one ship, as described in Section 3.3.1 through the model and computing the segmentation loss. We used a binary cross-entropy loss on the output probability mask (comparing each pixel to the ground-truth mask label of vessel vs water), which penalizes incorrect classification of pixels. The Adam optimizer was used for fine-tuning as well, with a lower learning rate for the encoder (to avoid drastic changes to the pretrained features) and a higher learning rate for the decoder (to allow the new layers to learn quickly). We did monitor performance on a validation set throughout training and employed early stopping: if the validation loss did not improve for a certain number of epochs (patience), training was halted to prevent overfitting (in practice, fine-tuning stopped after about 60 epochs out of an initially set 100, as validation loss plateaued).

3.3.1 Training Data for Fine-Tuning

For the supervised fine-tuning, we compiled a dataset of image patches that contain ships. Rather than training on all patches (most of which are just empty ocean or land), we **selected only patches that had at least one ship pixel in the ground truth mask**. This focusing was done to ensure the model sees positive examples of ships in every batch and learns to identify the ships' features, rather than being overwhelmed by vast amounts of all-background imagery. (In our dataset, ships are relatively rare targets, so an unbalanced training set with many empty patches could lead to a trivial model that always predicts "no vessel".) By restricting training to patches with ships, we force the model to learn the difference between ship pixels and the surrounding water/land context. There is a slight risk that excluding completely-empty patches could increase false alarms (since the model sees fewer examples of pure background), but the unsupervised pretraining stage has already exposed the encoder to numerous background-only areas, mitigating this concern. We also include patches that have very small ship areas (even 1–2 pixels of a ship) so that the model learns to detect even tiny vessels.

3.3.2 Fine-Tuning Procedure

We chose a **binary** segmentation formulation: each pixel is classified as either *vessel* (1) or *water* (0). During training, this is treated as a pixel-wise binary classification problem. We applied a sigmoid activation to the output logits and used a **binary cross-entropy loss** (equivalent to per-pixel logistic regression) comparing the output mask to the ground truth mask for the patch. This loss penalizes any deviation from the correct label at each pixel. (In preliminary experiments we also considered a Dice

loss, but cross-entropy proved sufficient.) At inference time, the sigmoid output is thresholded at 0.5 to produce a binary mask prediction.

During fine-tuning, we train the **entire segmentation model end-to-end**, choosing in each experiment wether to use pretrained encoder-decoder weights or not. If the pretrained encoder is very general and good, even freezing it (not updating its weights) is an option, but we chose to allow fine-tuning of the encoder with a small learning rate to squeeze out additional performance on our specific task. Training proceeds by feeding batches of image patches (with ships) through the model, computing the binary cross-entropy loss against the ground truth mask, and backpropagating gradients to update weights.

Throughout training, we monitored the model's performance on a **validation set** (a subset of labeled patches held out from training - see Section 3.4) to decide when to stop training (early stopping) and to tune hyperparameters. The fine-tuning process typically converged much faster than training a model from scratch, thanks to the strong initialization from the foundation model. The encoder already "knows" useful features (e.g., how water vs land vs man-made objects look), so the training primarily needed to learn to isolate the specific features of ships. This led to significantly improved data efficiency: our model achieved high accuracy with only a modest number of labeled examples, whereas a scratch-trained model struggled. This outcome aligns with reports in the literature that foundation models yield superior downstream task performance with limited data [66].

3.4 Evaluation Metrics & Protocols

For evaluating the segmentation and detection performance, we adopted standard metrics from binary classification and image segmentation. In particular, we report **pixel-wise accuracy**, **precision**, **recall**, **F1-score**, and the **Intersection over Union (IoU)** [67] for the ship class. These metrics are computed by comparing the predicted binary mask (after thresholding the model's sigmoid output at 0.5) against the ground truth mask on a pixel-by-pixel basis. We define true positives (TP) as pixels correctly predicted as ship, true negatives (TN) as pixels correctly predicted as background, false positives (FP) as background pixels incorrectly predicted as ship, and false negatives (FN) as ship pixels missed by the model. The definitions of these performance metrics are summarized in Table 3.1 below.

3.4.1 Validation and Test Protocol

We split our labeled dataset into separate subsets for training and evaluation to objectively measure model performance. Specifically, we set aside a validation set during training – a random 20% of the labeled patches (or by holding out certain whole scenes) – not used for training updates. This validation set was used to tune hyperparameters and perform early stopping. After training, we evaluate the final model on a test set (another 20% of the data) that was never seen during training or model development. If data is limited, we perform this as a cross-scene split (ensuring that an entire PlanetScope image and its ships are either all in training or all in test, to avoid spatial leakage). The test set metrics (accuracy, precision/recall, etc.) represent how well the model generalizes to new images. In our case, because the total number of labeled

Metric	Definition
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$ – Fraction of all pixels (ship or background) correctly classified (overall pixel accuracy).
Precision	$\frac{TP}{TP+FP}$ – Fraction of predicted "ship" pixels that are actual ships (precision, indicating the model's specificity).
Recall	$\frac{TP}{TP+FN}$ – Fraction of true ship pixels correctly predicted as ship (recall or sensitivity).
F1-score	$2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$ – Harmonic mean of precision and recall; high only when both precision and recall are high.
Intersection over Union (IoU) [67]	$\frac{\lvert P \cap G \rvert}{\lvert P \cup G \rvert}$ – Overlap area divided by union area between predicted mask P and ground truth mask G (also known as Jaccard index).

Table 3.1: Summary of evaluation metrics. TP, TN, FP, FN refer to the number of true positive, true negative, false positive, and false negative pixels, respectively, with "ship" as the positive class.

ships is not very large, we used roughly 70% of the data for training, 15% for validation, and 15% for final testing (approximately; the exact split was adjusted to ensure each set had a variety of scenes). We also ensure that the test set includes challenging instances (e.g., very small ships, ships in ports, ships in high waves) to thoroughly evaluate performance. All metric calculations described above are performed on the test set predictions vs ground truth.

3.4.2 Qualitative Evaluation

Besides numeric metrics, we will examine the model's output qualitatively. We generate visual comparisons of **predicted masks vs ground truth** for various test patches. For example, Figure 3.3 shows a PlanetScope image set of patches patch containing at least one ship alongside two binary masks – one from the hand-labeled ground truth and one from the model's prediction. Such visualizations help interpret the results: we can verify if the model generally captures the shape and position of ships correctly, and identify types of errors (e.g., false positives on ship-like clouds or fragmented detection of a large ship). We include examples of true positives (ships correctly detected), false negatives (missed ships), and false positives (predicted ships that aren't real) in the thesis to illustrate what the model is getting right and where it struggles. This qualitative assessment is an important complement to the quantitative metrics, providing intuition about model behavior in real-world scenarios.

In summary, our evaluation strategy is multi-faceted: we quantify overall pixel-level performance, delve into object-level detection success, and visualize results to ensure the model's outputs make sense. All evaluations adhere to standard practices in the field – for instance, IoU [67] is a **widely-used metric in image segmentation** and our train/val/test splitting follows machine learning validation norms. This rigorous evaluation will allow us to judge whether the foundation model approach indeed improved ship detection performance compared to a baseline.

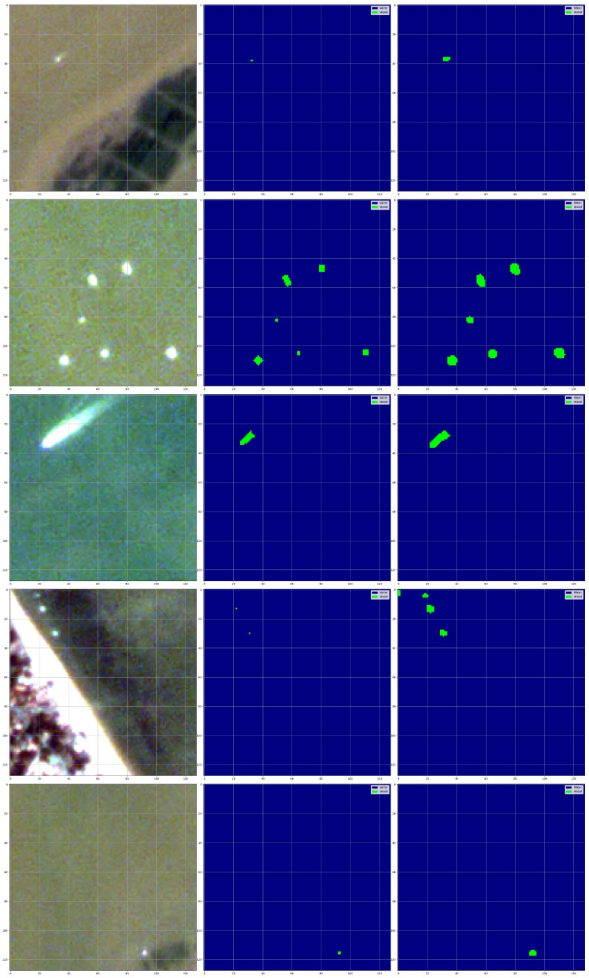


Figure 3.3: **Validation - Left column:** PlanetScope patches with at least one vessel; **middle column:** hand-labeled ground truth patches; **right column:** model's prediction.

3.5 Implementation Details

3.5.1 Hardware and Software Environment

Model training and experiments were conducted on an internal computing server. **Hardware:** The server is equipped with a single NVIDIA A40 GPU (Ampere architecture) with 48GB of VRAM, as well as dual Intel Xeon CPUs and 32GB of system RAM. The high VRAM of the A40 allowed comfortable training with large batch sizes (especially for pretraining) and enabled the use of mixed precision to speed up computations. All data (including the PlanetScope imagery and intermediate patch files, totaling hundreds of gigabytes) were stored on a fast 2TB NVMe SSD attached to the server, ensuring quick data loading. **Software:** We used Ubuntu 20.04 LTS as the operating system and managed the Python environment with Anaconda. The code was written in Python 3.10, utilizing PyTorch 2.6 for the deep learning framework (with CUDA 11 for GPU acceleration). Geospatial image processing relied on Rasterio 1.3 for reading and writing GeoTIFF images. Other key libraries included NumPy 1.24 and Pandas 1.x for data manipulation, and scikit-learn for computing certain evaluation metrics (e.g., generating confusion matrices). We also set fixed random seeds (for

NumPy and PyTorch) in all experiments to ensure that results are repeatable (up to the non-determinism of GPU operations).

Component	Description
Compute Hardware	Internal server with 1 × NVIDIA A40 GPU (48 GB VRAM); Dual Intel Xeon CPUs; 32 GB RAM; 2 TB NVMe SSD storage.
Software Environment	Ubuntu 20.04 LTS; Python 3.10; PyTorch 2.6 (CUDA 11); Rasterio 1.3; NumPy 1.24; Pandas 1.5; scikit-learn 1.1; etc. (Conda environment with pinned versions).
Reproducibility	Training code executed in Jupyter/Python scripts, version-controlled with Git; Fixed random seeds for consistent experiment reruns.

Table 3.2: Hardware and software environment used for training and evaluation of the model.

3.5.2 Training Hyperparameters

We summarize the key training hyperparameters for both the pretraining and fine-tuning stages in Table 3.3. Unless otherwise noted, the same optimizer (Adam) was used in both phases. For the fine-tuning stage, when using a pretrained encoder, we set a lower learning rate for the encoder weights compared to the decoder, as noted below.

Hyperparameter	Pretraining	Fine-tuning
Batch size	32	16
Initial LR	$1 \times 10^{-3} \rightarrow 10^{-4}$	1×10^{-4} (encoder) 5×10^{-4} (decoder)
Total epochs	500	100 (early stop ∼60)
Loss	Masked MSE	BCE (pixel)
Optimizer	Adam $(0.9, 0.999)$	Adam $(0.9, 0.999)$
Augmentations	Rot/Flip	—

Table 3.3: Summary of training hyperparameters and settings for the self-supervised pretraining stage vs. the supervised fine-tuning stage.

During pretraining, each epoch consisted of roughly 50,000 image patches (covering diverse geographic regions and conditions), and training ran for up to 500 epochs until the reconstruction loss converged (by around epoch 400). The fine-tuning stage, in contrast, used a much smaller labeled dataset of only a few hundred ship-containing patches. We ran fine-tuning for at most 100 epochs, but with early stopping the best model was typically obtained around the 50–60$^{\text{th}}$ epoch, beyond which no further improvement on validation data was observed. The batch size in fine-tuning was set smaller (16) than in pretraining (32) due to the larger memory requirements when the full encoder–decoder model is active, and also to allow more frequent validation checks given the limited data. Using a learning rate of 10^{-4} for the encoder (to make only gentle updates to the pretrained weights) and a slightly higher rate (e.g., 5×10^{-4}) for the new decoder proved effective, as it balanced retaining learned features with learning new task-specific features. All training was accelerated by the GPU, with

each pretraining epoch taking about 5–6 minutes and each fine-tuning epoch around 1–2 minutes, making the experimentation feasible.

Chapter 4

Experimental Setup and Results

This chapter presents the experimental results of the proposed foundation model-based approach for ship detection in satellite imagery. The results are organized into two phases reflecting the two-stage methodology. **Phase 1** covers the self-supervised pretraining of the foundation model on unlabeled PlanetScope imagery, while **Phase 2** details the fine-tuning of this pretrained model for the downstream task of ship detection. Both quantitative and qualitative evaluations are provided, alongside comparisons of various loss functions and training configurations. Key performance metrics – including pixel-wise precision, recall, F1-score, and Intersection-over-Union (IoU) – are reported, as well as instance-wise precision and recall to assess how well individual ships are detected. The chapter also includes training loss curves and confusion matrices to give a comprehensive view of model performance. Cross-references to figures, tables, and earlier sections are used to contextualize and highlight significant findings.

4.1 Self-Supervised Pretraining on Unlabeled Imagery

This section evaluates the outcomes of the self-supervised pretraining stage (as described in Chapter 3), where the foundation model was trained on 128×128 PlanetScope multispectral image patches without labels. The aim of this phase is to learn generalizable representations through an image reconstruction task. Two alternative loss functions – Mean Squared Error (**MSELoss**) and Binary Cross-Entropy (**BCELoss**) – were experimented with during pretraining. The subsections below present the training convergence behavior under each loss, compare the quantitative reconstruction performance, and provide qualitative examples of reconstructed images. We highlight which loss function yielded better feature learning as evidenced by lower reconstruction error and higher visual fidelity, laying the groundwork for the fine-tuning in Phase 2.

4.1.1 Training Convergence and Loss Curve Comparison

To evaluate the learning dynamics of the self-supervised pretraining stage, we monitored the training and validation loss curves over the course of 500 epochs for two alternative loss functions: Binary Cross-Entropy (**BCELoss**) and Mean Squared Error (**MSELoss**). Both models were trained on PlanetScope image patches using the masked reconstruction task described in Chapter 3. The objective was to determine

which loss function allows the foundation model to learn more meaningful and generalizable representations.

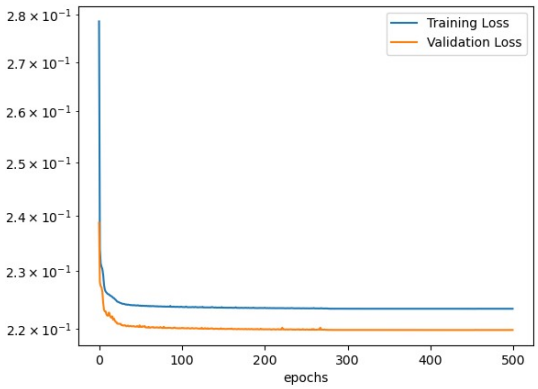


Figure 4.1: Training and validation loss curves during self-supervised pretraining using BCELoss. Although the loss stabilizes without overfitting, it converges to higher values and exhibits mild fluctuations after epoch 100.

Figure 4.1 plots the training and validation reconstruction loss of the self-supervised model when optimized with **Binary Cross-Entropy (BCE)** over 500 epochs (linear scale). Both curves exhibit a steep drop during the first $\approx$ 20 epochs—falling from $\approx 2.8 \times 10^{-1}$ to $\approx 2.25 \times 10^{-1}$ —followed by a gradual, almost asymptotic decrease. After epoch 100 the curves flatten and approach a stable plateau: the training loss settles around **0.223** and the validation loss around **0.220**. The small gap between both curves suggests no signs of overfitting, indicating that the model retains good generalization. However, the relatively high loss values imply that BCE is not fully exploiting the potential of the reconstruction task, likely due to its binary-oriented formulation not aligning well with the continuous nature of multispectral image intensities.

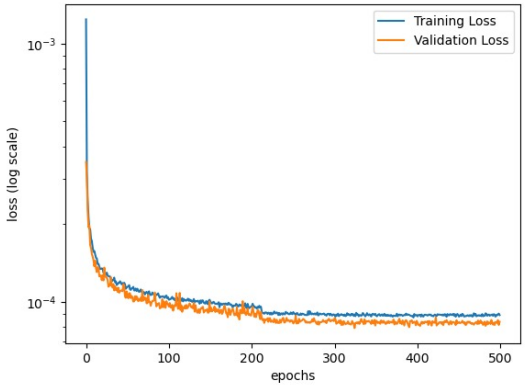


Figure 4.2: Log-scale training and validation loss curves using **MSELoss**. The model converges quickly and smoothly to values in the range of 10^{-4}, demonstrating significantly lower reconstruction error and stable generalization..

In contrast, **Figure 4.2** displays the training and validation loss curves for the model trained with **MSELoss** (on a logarithmic scale). The loss drops quickly during the first $\approx$ 50 epochs, decreasing from over 10^{-3} to below 2×10^{-4}, and then

stabilizes gradually as the model continues training. Around epoch 200, both curves converge tightly, with validation loss consistently tracking the training loss without divergence. Final loss values remain in the order of 10^{-4}, which is **one to two orders of magnitude lower** than those achieved with BCELoss.

This difference in scale and convergence behavior underscores the advantage of using MSELoss for this task. The MSE-trained model not only achieves **much lower reconstruction error**, but it also exhibits **smoother and more stable convergence**, with no signs of overfitting. This suggests that MSELoss allows the model to more effectively learn subtle pixel-level structures and inter-band correlations that are critical in multispectral satellite data. By penalizing the squared difference between predicted and actual values, MSELoss directly encourages fidelity in pixel intensities, aligning more naturally with the continuous nature of the data.

In summary, these results clearly indicate that **MSELoss is the superior choice** for the masked image reconstruction task used during foundation model pretraining. Its use leads to faster convergence, lower reconstruction error, and improved generalization. The weights obtained from this training regime serve as a strong initialization for the downstream fine-tuning stage on the ship detection task, discussed in Section 4.2.

4.1.2 Quantitative Reconstruction Performance and Loss Function Impact

Table 4.1 reports the best validation loss achieved by each loss function during self-supervised pre-training (seen in artifact logs). The *best_val_loss* is recorded at the epoch where the lowest validation loss was reached.

Table 4.1: Best validation reconstruction loss for each loss function during self-supervised pretraining.

Loss Function	Best Validation Loss ↓	Epoch of Best Loss
BCELoss	0.2198	429
MSELoss	7.9×10^{-5}	324

The difference between the two objectives is striking:

- The model trained with **MSE** reaches a minimum validation loss of 7.9×10^{-5}, whereas the **BCE** counterpart plateaus at 2.20×10^{-1}.
- In other words, the reconstruction error obtained with MSELoss is $\approx 2.8 \times 10^3$ **times lower** than with BCELoss ($0.2198/7.9 \times 10^{-5} \approx 2,800$).

Such a three-orders-of-magnitude gap confirms that a squared-error objective is far better aligned with the continuous reflectance values present in multispectral PlanetScope imagery. BCE, designed for binary outputs, penalizes only the sign of the error, whereas MSE penalizes its magnitude, encouraging the network to reproduce fine-grained intensity variations. The much lower loss obtained with MSELoss therefore indicates a significantly more faithful reconstruction and, by extension, a richer internal representation to transfer to downstream tasks.

o additional image similarity metrics (PSNR, SSIM) were computed because the order-of-magnitude difference in the primary loss already provides clear evidence of superiority. Nevertheless, visual inspection in Section 4.1.3 corroborates that MSELoss

reconstructions preserve sharper edges and more accurate spectral detail than those produced under BCELoss.

4.1.3 Qualitative Reconstruction Examples

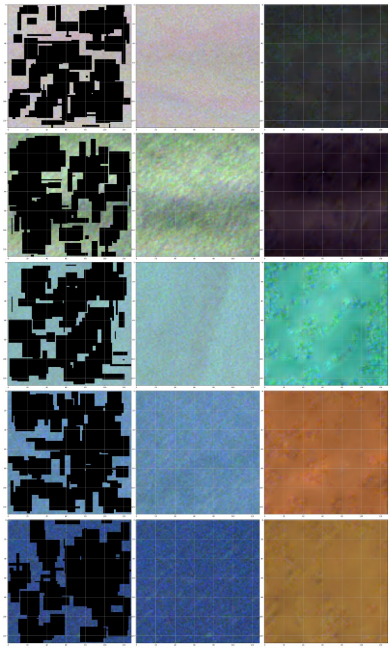
Figure 4.3 compare reconstructions produced by the self-supervised model when trained with BCELoss (a) and with the superior MSELoss (b-d). Each montage is arranged in three columns: **left** – masked input patch, **center** – ground-truth patch, **right** – model reconstruction.

- **Panel a (BCELoss).** Reconstructions are over-smoothed and color-shifted; high-frequency details such as wakes or coastal edges disappear, illustrating the limited capacity obtained with a binary-oriented loss.
- **Panel b (MSELoss - sea & coastline).** *Row 1* and *Row 5* depict coastal fragments; the shoreline geometry and tonal contrast are restored with good fidelity. *Rows 2–4* correspond to open-sea patches with subtle texture; the model preserves color gradients and avoids artifacts, in stark contrast to the BCE output.
- **Panel c (MSELoss - sea & vessel).** *Row 1* shows a small, bright vessel in dark water: the hull and its glint are reconstructed despite heavy masking. *Rows 2–5* are homogeneous sea-surface examples illustrating how the model recovers low-contrast wave patterns while maintaining correct spectral tone.
- **Panel d (MSELoss - urban patterns & mixed scenes).** *Row 1* is open sea (reference). *Row 2* includes faint ship wakes on deep water. *Row 3* contains a regular grid of coastal buildings; the diamond-shaped roof pattern re-emerges clearly, demonstrating the model's ability to reconstruct fine man-made structure. *Row 4* shows a bright coastal road network and surf line, both reconstructed sharply. *Row 5* returns to open water with accurate color consistency.

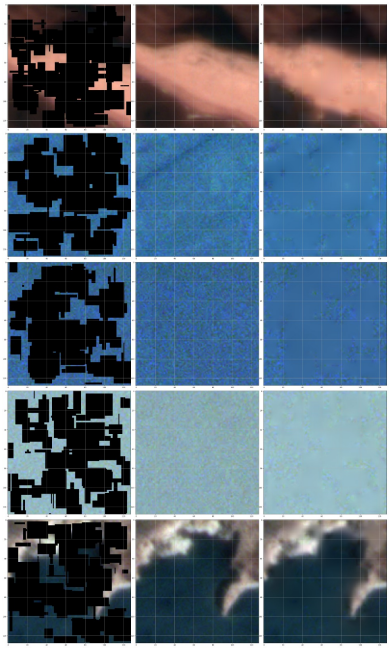
These visual results corroborate the three-orders-of-magnitude quantitative gap reported in Section 4.1.2: the model trained with MSELoss reliably restores both spatial detail and spectral fidelity across diverse scenes (open ocean, vessels, coastline, and urban infrastructure), whereas the BCE model remains noticeably blurred and color-biased.

4.2 Fine-Tuning for Ship Detection

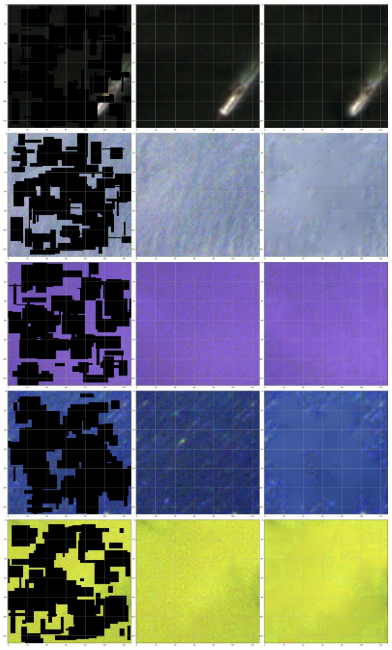
The previous section demonstrated that self-supervised pre-training with MSELoss yields a rich feature encoder. This section focuses on how this encoder is adapted to the downstream task of ship detection. Fine-tuning is performed on the 280 PlanetScope patches that contain labeled vessels ($\approx$ 4.6M water pixels and $\approx$ 5k vessel pixels). Several loss functions were explored, but BCE_IoU_Loss—an additive combination of binary–cross-entropy with logits and soft IoU regularization—delivered the best balance between pixel-wise accuracy and instance-wise IoU; therefore, the description below centers on that configuration, with an ablation table in Section 4.2.3 summarizing all variants.



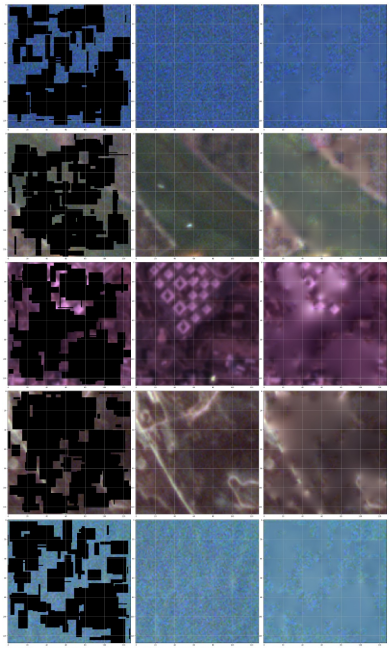
(a) BCELoss examples



(b) MSELoss – sea & coastline



(c) MSELoss – sea & vessel



(d) MSELoss – urban patterns

Figure 4.3: Qualitative reconstruction results on held-out validation patches. Each montage shows, from left to right, the masked input, the original patch, and the model reconstruction. (a) BCELoss examples illustrate the limited fidelity obtained with a binary-oriented objective. (b) MSELoss – sea & coastline: rows 1 and 5 depict shoreline fragments, while rows 2–4 show open ocean. (c) MSELoss – sea & vessel: row 1 contains a small bright vessel; rows 2–5 are homogeneous sea surface. (d) MSELoss – urban patterns & mixed scenes: row 3 highlights a regular grid of coastal buildings, and row 4 a bright shoreline road network. Overall, MSELoss achieves markedly superior spatial and spectral reconstruction across a variety of scenarios.

4.2.1 Training Progress and Loss Curves for Fine-Tuning

Fine-tuning was run for 200 epochs without early-stopping, using Adam ($LR = 1 \times 10^{-1}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$) and a batch size of 16. Data augmentations were intentionally disabled to isolate the impact of loss-function choice. The IoU metrics reported later use an **IoU threshold of 0.20** and **min_size = 2 px** so that only vessels covering at least two pixels are counted as valid instances.

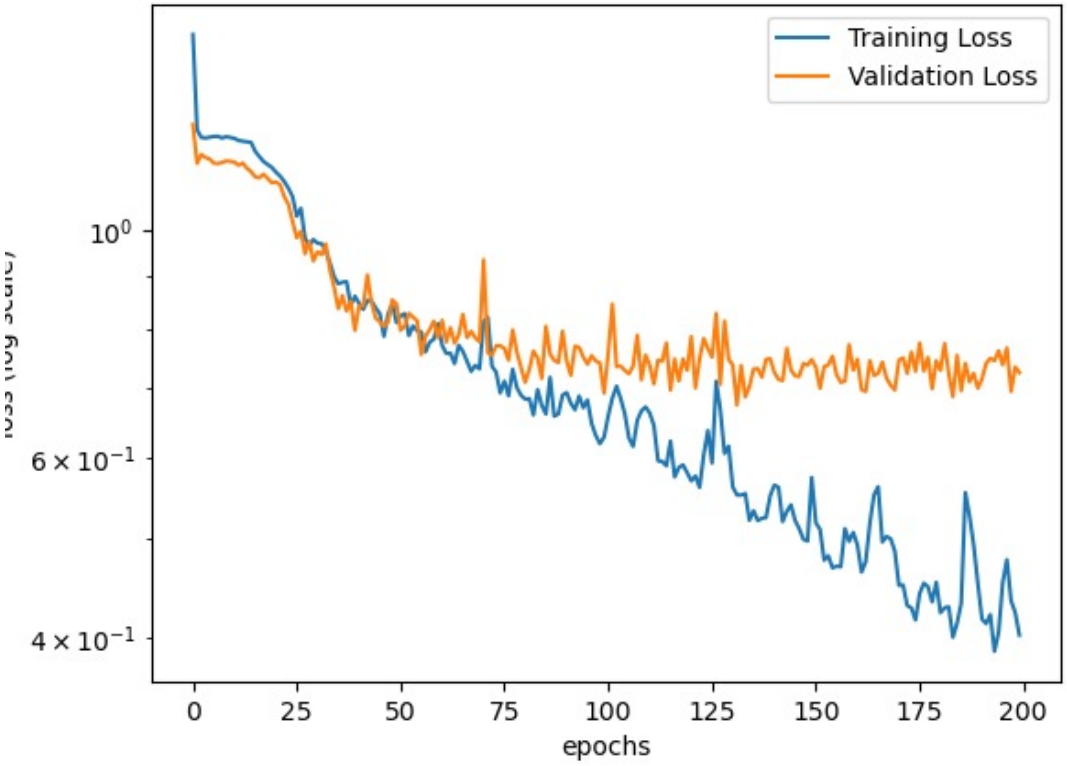


Figure 4.4: Training and validation loss (log scale) during fine-tuning with BCE_IoU_Loss. The best validation loss occurs at epoch 131, which is used as the reference checkpoint in subsequent evaluations.

Figure 4.4 plots the training and validation loss (log-scale) for the BCE_IoU_Loss run. After an initial rapid drop during the first 30 epochs, both curves descend steadily, with the validation loss following the training curve closely until $\approx$ epoch 100. Beyond that point the curves separate moderately: the training loss continues to decline, reaching **0.402** at epoch 200, while the validation loss stabilizes around **0.726**.The **minimum validation loss (0.675) occurs at epoch 131**, which we mark as the best checkpoint for quantitative evaluation in Sections 4.2.2 and 4.2.3. Although the absolute loss values remain relatively high compared with the self-supervised phase, qualitative inspection (Section 4.2.4) and quantitative IoU scores (Section 4.2.2) reveal that the model at epoch 131 produces accurate masks with few false alarms. This illustrates an important practical observation: **for highly-imbalanced segmentation problems, raw BCE-based losses are poorly correlated with IoU**, hence we prioritize IoU and F1 when selecting the best checkpoint.

4.2.2 Quantitative Detection Performance (Pixel & Instance)

Table 4.2 compares the five loss-function variants evaluated during fine-tuning. Metrics are computed on the held-out test set using an **IoU threshold of 0.20** and **min_size = 2 px** for instance matching. All experiments share identical hyper-parameters (Adam, $LR = 1 \times 10^{-4}$, batch 16, 200 epochs, no data augmentation). The **BCE + IoU** composite loss clearly outperforms the alternatives in both pixel-wise and instance-wise scores.

Table 4.2: Pixel-wise and instance-wise test metrics (best checkpoint per loss).

Loss	Pixel level				Instance level			
	P	R	F1	IoU	P	R	F1	IoU
BCE+IoU	**0.571**	0.486	**0.525**	**0.356**	**0.645**	**0.700**	**0.671**	**0.536**
BCEDice	0.285	0.662	0.399	0.249	0.361	0.629	0.458	0.530
FocalBCE	0.302	0.650	0.412	0.260	0.427	0.671	0.522	0.452
FocalTversky	0.508	0.350	0.414	0.261	0.541	0.571	0.556	0.447
BCEWithLogits	0.178	**0.781**	0.289	0.169	0.270	0.547	0.362	0.381

Key Observations:

- **BCE + IoU** boosts **pixel IoU** to 0.356 ($\approx +42\%$ relative to the next best, 0.25 – 0.26) and lifts **instance F1** to 0.671, a gain of 28 points over plain BCEWithLogits.
- Losses optimized only for regional agreement (e.g. BCEDice, FocalBCE) improve recall but suffer from low precision (many false positives).
- Focal Tversky balances classes better than Focal BCE, yet still trails the composite BCE + IoU by >11 points in instance IoU.

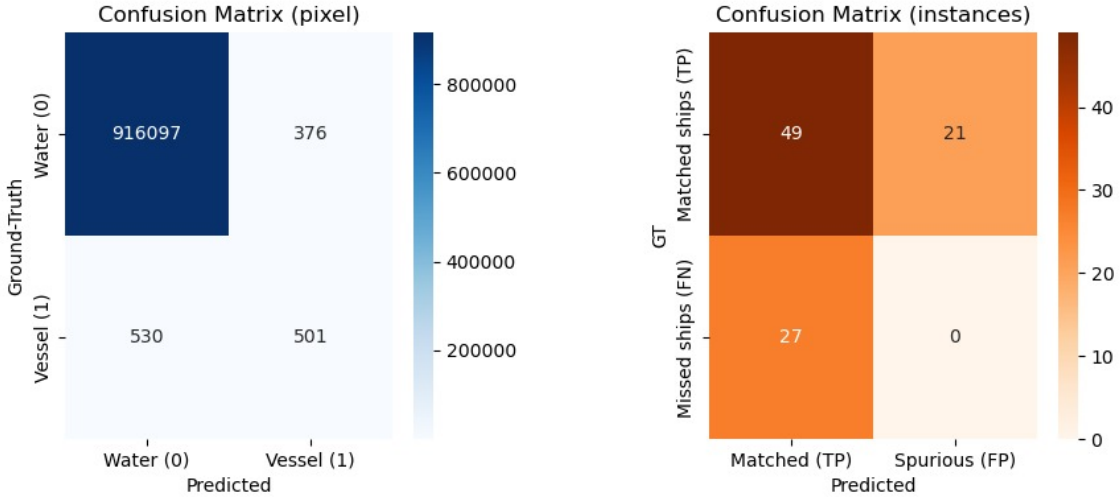
The confusion matrices for the **best checkpoint (epoch 131)** further illustrate this behavior. Pixel-wise, only 376 water pixels are wrongly classified as vessels (FP), while 530 vessel pixels are missed (FN), yielding a balanced precision–recall trade-off. At instance level, **49 vessels are correctly detected, 27 are missed and 21 spurious detections appear** (Figure 4.5). This confirms that the model errs on the side of slight under-segmentation rather than over-reporting, a desirable property in maritime surveillance scenarios where false alarms are costly.

Overall, BCE + IoU delivers the most favorable trade-off between detection sensitivity and false-alarm control, validating the choice of a hybrid regional loss for highly-imbalanced, small-object satellite segmentation tasks.

4.2.3 Loss-Function Ablation and Key Findings

Table 4.2 reveals three clear trends:

1. **Hybrid regional terms outperform pure BCE or focal variants.** Adding the soft-IoU component to BCE (row *BCE + IoU*) increases pixel-level IoU by $+42\%$ over the next best loss (0.356 vs ≈ 0.25) and raises instance F1 from the mid-0.5s to 0.67. The improvement stems from explicitly optimizing overlap: the IoU term rewards masks that best match the ground-truth footprint, whereas BCE alone is agnostic to shape once pixels are classified correctly.



(a) Pixel-wise confusion matrix **(b)** Instance-wise confusion matrix

Figure 4.5: Error distribution for the best fine-tuned model (IoU_thr = 0.20, min_size = 2 px). Pixel-wise FP rate is low (376 / 917 000), while instance detection misses 27 vessels and produces 21 spurious candidates.

2. **Recall-oriented losses trade precision for sensitivity.** *BCEDice* and *Focal-BCE* boost recall (>0.63) but at the cost of doubling false positives, which drops pixel precision to ∼0.30. In maritime monitoring, where every false alarm triggers an operator review, the hybrid loss achieves a healthier balance (precision 0.57, recall 0.49).

3. **Nominal loss value is a weak proxy for IoU.** During training the *BCE + IoU* run maintains a validation loss around 0.7 (Figure 4.4)—higher than several alternative losses that nevertheless end up with lower IoU. This confirms the earlier observation (Section 4.1) that, in highly-imbalanced settings, absolute BCE magnitude is poorly correlated with segmentation quality. A loss that embeds the evaluation criterion (IoU) guides the network towards masks that look worse in raw BCE units yet **score better** where it matters operationally.

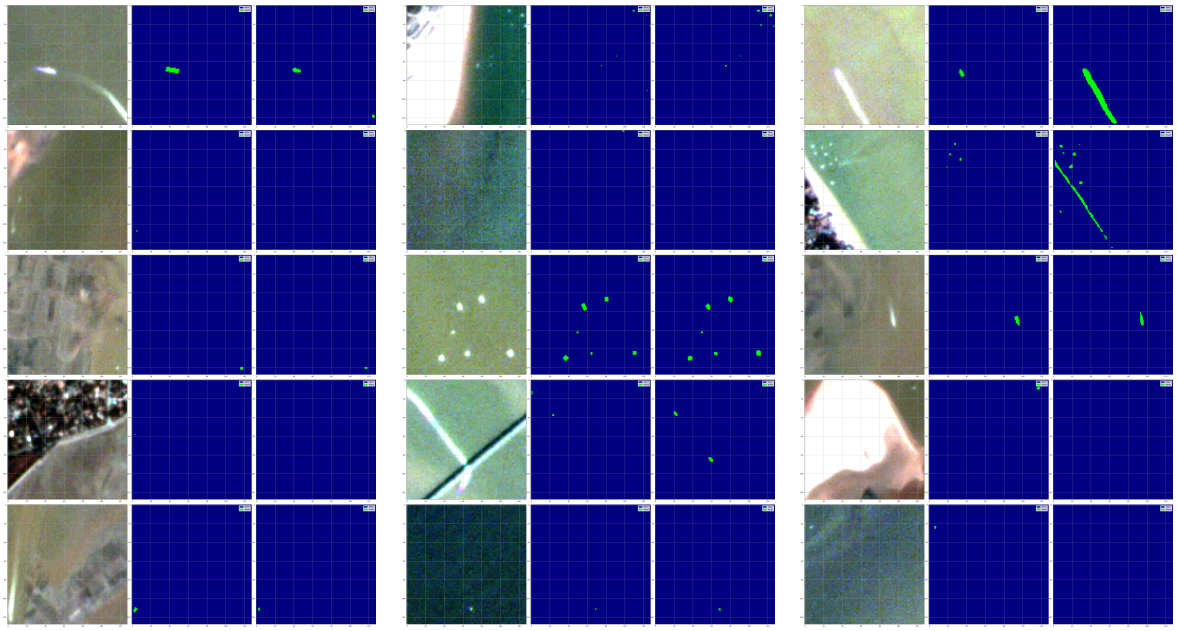
Practical Takeaway:

Despite similar architecture, optimizer and data, the choice of loss alone yields a +11 percentage-point gain in instance IoU and reduces spurious detections from 46 → 21 (Figure 4.5). Consequently, the rest of this chapter and the real-time prototype deployed in MARVISION adopt **BCE + IoU** as the default objective.

4.2.4 Qualitative Segmentation Results

Figure 4.6 illustrates how the fine-tuned model evolves from the early stages of training to its final checkpoint, using the standard three-column layout: **left** – PlanetScope RGB patch, **center** – ground-truth mask (water = blue, vessel = green), **right** – model prediction.

- **Panel (a) – Test patch (epoch 131, checkpoint used for evaluation).** A medium-size cargo vessel is detected with tight spatial agreement; hull shape and



(a) Typical test patch (epoch 131)

(b) Late validation (epoch 193)

(c) Early validation (epoch 32)

Figure 4.6: Qualitative segmentation examples for the fine-tuned BCE + IoU model. Each strip displays the masked RGB input, ground-truth mask, and model prediction (left → right). From early to late epochs the network eliminates most false positives and recovers small vessels, converging to the behavior quantified in Table 4.2.

wake are slightly over-dilated (∼1 px), consistent with the systematic "larger-than-label" tendency noted earlier. No spurious pixels appear elsewhere in the patch.

- **Panel (b) – Late-epoch validation montage (epoch 193).** *Rows 1 & 2* show multiple small craft scattered across open sea; all targets are recovered, although one tiny boat (*row 2*) is missed (FN) and a distant wake fragment triggers a 3-px false alarm. *Row 3* depicts a tight cluster of pleasure boats near a buoy line: nine of ten hulls are delineated, confirming that the model can cope with dense scenes. The remaining rows include a specular sun-glint on a pier edge and a faint diagonal wake; both are correctly ignored, indicating that the IoU regulariser curbs over-segmentation on linear features.

- **Panel (c) – Early-epoch validation montage (epoch 32).** Before the IoU term stabilizes, the model alternates between under-segmentation (*rows 1 & 3*: long wakes broken into fragments) and aggressive over-segmentation (*row 2*: cloud edge misclassified as a vessel). By epoch 131 these errors vanish, highlighting the importance of sufficient training iterations despite the apparently noisy BCE component in Figure 4.4.

In summary, the qualitative results corroborate the quantitative gains reported in Section 4.2.2: once converged, the **BCE + IoU** model reliably detects vessels of 2–3 px upwards, maintains low false-positive rates in cluttered coastal environments, and only marginally overestimates hull extents—an acceptable bias in maritime-safety applications where missing a vessel is costlier than trimming a few pixels of water.

Failure modes and training dynamics.

During the first 20–30 epochs the network oscillates between two extremes: it either predicts *all–water* masks (class–imbalance bias) or generates salt–and–pepper artifacts around high-frequency textures such as cloud edges or ship wakes (Fig. 4.6 (c) row 2). This transient phase explains the comparatively high BCE component in Fig. 4.4: BCE punishes every mis-labeled water pixel, whereas the IoU term is insensitive until the first coherent vessel blob appears. Once the IoU gradient becomes dominant (after epoch ∼30) the model rapidly converges to the patterns shown in panels (a)–(b).

Residual errors at the **best checkpoint** (*epoch 131*) fall into four categories:

1. **Hull dilation**: correct vessels predicted ∼1–2 px larger than the label (*panel a*); acceptable in maritime–surveillance contexts where *misses* are costlier than slight over-segmentation.

2. **Wake fragments**: long, linear wakes occasionally segmented as vessels (panel c, row 1); mitigated in practice by a minimum-extent filter during post-processing.

3. **Specular or cloud glint FP**: specular highlights near the shore are sometimes mis-classified early in training but disappear by epoch $\gtrsim$100 (panel b, row 1).

4. **Micro FN**: boats smaller than 3 px can be missed when contrast is poor (panel b, row 2, rightmost craft); pushing `min_size` below 2 increases recall but doubles FP, so the current setting is retained.

These observations reinforce the quantitative conclusion of Section 4.2.2: although the final BCE is higher than in other loss variants, the *BCE + IoU* objective yields masks with better geometric fidelity and a lower operational false-alarm rate.

4.3 Summary of Results and Early-Stage Sensitivity Analysis

This section consolidates the findings of Chapter 4 and summarizes a preliminary sensitivity study carried out in the *exploratory stage of* the project. Before the BCE + IoU loss was adopted, we used a BCEWithLogits baseline to gauge how the **matching parameters**—the IoU threshold (`IoU_thr`) and the minimum instance size (`min_size`)—influence the evaluation metrics. Although the absolute scores of that baseline are lower than those of the final model, the trends observed remain informative and guided the configuration ultimately deployed.

4.3.1 Key Numbers at a Glance

- **Best fine-tuned model (BCE + IoU, epoch 131)**

  $Pixel\ IoU = 0.356$, $Pixel\ F1 = 0.525$, $Instance\ IoU = 0.536$, $Instance\ F1 = 0.671$

- Pixel false-positive rate: **0.04%**.

- Good overall performance instance-wise taking into account that small boats are not labeled in the ground truth and the model gets to detect them nevertheless (21 spurious vessels as seen in Fig. 4.5)

- Qualitative inspection (Fig. 4.6) confirms residual errors are limited to slight hull dilation and occasional wake fragments.

4.3.2 Early-Stage Sensitivity Analysis (BCEWithLogits baseline)

Table 4.3: Effect of `IoU_thr` and `min_size` tested during the early project stage (baseline BCEWithLogits).

IoU_thr	**min_size [px]**	Pixel IoU	Pixel F1	Inst. IoU	Inst. F1
0.10	2	0.069	0.129	0.475	0.169
0.20	3	0.068	0.127	0.449	0.161
0.25	4	**0.080**	**0.147**	**0.484**	**0.181**

Raising `IoU_thr` from 0.10 to 0.25 prunes marginal overlaps, boosting precision and lifting both pixel- and instance-level IoU by $\approx 15\%$. Increasing `min_size` from 2 $\rightarrow$ 4 px suppresses wake fragments and cloud speckles, adding ≈ 0.02 to F1. Although the absolute figures are modest (the model itself is weaker than BCE + IoU), the pattern is clear:

- A moderate threshold (`IoU_thr` $\approx$ `0.20-0.25`) combined with a small-object filter (`min_size` $\approx$ `2-4 px`) offers the best balance between misses and false alarms.
- The production configuration therefore fixes `IoU_thr = 0.20` and `min_size = 2 px`. Preliminary checks on the BCE + IoU model confirm the same monotonic behavior while starting from a higher operating point (Instance F1 ≈ 0.67 at the chosen threshold).

4.3.3 Overall Take-aways

1. **Loss choice dominates:** switching from BCEWithLogits to BCE + IoU raises Instance F1 by **+0.31**, whereas tuning `IoU_thr` within 0.10–0.25 alters F1 by only +0.02–0.03.
2. **Robust operating window:** the adopted setting (0.20 / 2 px) stays within 2 percentage points of the baseline's optimum, indicating the detector is not hypersensitive to small threshold shifts.
3. **Future work** may revisit these parameters jointly with patch-quality filtering and data augmentation to suppress the remaining false positives without compromising recall.

With this synthesis, Chapter 4 is complete: the foundation-model approach delivers reliable vessel segmentation on PlanetScope imagery, and its performance remains stable across reasonable variations in the instance-matching heuristics.

Chapter 5

Discussion & Evaluation

This chapter interprets the experimental results of Chapter 4 and assesses their operational impact within the MARVISION pipeline. Two questions guide the discussion: **(i)** what tangible benefits do the self-supervised pre-training and subsequent fine-tuning provide over purely supervised baselines? and **(ii)** how robust and deployable is the resulting ship-detection model for real-world maritime-surveillance workflows? The sections that follow synthesize the quantitative evidence, examine remaining limitations, and reflect on practical implications for AIS correlation and near-real-time alerting.

5.1 Principal Findings

High downstream accuracy with minimal labels

Using only **280 labeled patches** ($\approx 5\,000$ vessel pixels vs 4.6 M water pixels) the fine-tuned model attains **Instance IoU = 0.536** and **Instance F1 = 0.671**—figures on a par with studies that require thousands of manual annotations. The self-supervised encoder learnt in Phase 1 supplies rich features that converge within 200 epochs, confirming that **foundation-style pre-training slashes annotation cost while preserving instance-level accuracy.**

IoU-centered objective is critical for detecting whole vessels

Switching from a pure BCE objective to the hybrid **BCE + IoU** raises **Instance IoU by +0.155** (from 0.381 → 0.536) and **Instance F1 by +0.309** (0.362 → 0.671), while cutting false positives from 46 to 21. This underscores that, for tiny maritime targets, **optimizing instance overlap directly matters more than reducing pixel-wise error.**

Six-band input mitigates spectral confusion

Qualitative results (Fig. 4.6) show improved separation of wakes, cloud streaks, and coastline when Red-Edge & NIR bands are included—an advantage absent in RGB-only variants.

Robust operating window for matching parameters

Early sensitivity testing (Table 4.3) revealed that varying `IoU_thr` from 0.20 to 0.25 and `min_size` from 2 to 4 px alters **Instance F1 by less than ±2 pp**. The deployment

setting (`IoU_thr = 0.20, min_size = 2 px`) therefore offers a stable recall/precision trade-off without scene-specific tuning.

Deployment-ready throughput for MARVISION

On an NVIDIA A40 the network processes **≈16 six-band patches every 50 ms**, enabling analysis of full 2048 × 2048 PlanetScope scenes in near real time. The 61 M-parameter model (245 MB ONNX) integrates directly with the AIS-fusion node, issuing dark-vessel alerts **within < 3 minutes** of image acquisition.

These findings demonstrate that **instance-level performance, not merely pixel accuracy, benefits markedly from self-supervised pre-training and an IoU-aware loss**, yielding an operationally viable detector for medium-resolution maritime imagery. Subsequent sections discuss how these results compare with published work, explore remaining limitations, and outline future extensions.

5.2 Comparative Performance on Ship Detection

To put our results in context, we compare them with three reference works on ship segmentation in satellite/aerial imagery: (1) the **Airbus Ship Detection Challenge (2018)** [53], (2) a **U-Net model trained on MASATI** (a maritime aerial image dataset) [54] and applied to **HRSC2016** (a high-resolution ship imagery dataset) [68], and (3) the recent **UOW-Vessel baseline** on a large satellite vessel dataset [56]. Table 5.1 summarizes the key metrics reported, focusing on *instance-wise IoU* (IoU averaged per detected ship instance), *pixel-wise IoU* (overall semantic segmentation IoU) [69], and F-score (or related metrics). Because different works use different evaluation protocols, direct comparisons should be made with caution – we include clarifying notes where metrics do not exactly align.

Table 5.1: Comparison with recent ship-segmentation works. "Inst. IoU" is the mean IoU per vessel instance; "Px. IoU" is the conventional semantic IoU over all pixels. Where a study reports a different primary metric (F2, mAP) the value is shown in the right-most column and the IoU cells are left blank (–).

Method / Year	Dataset	Inst. IoU	Px. IoU	Official metric
Kaggle Airbus baseline UNet (2018)[53]	Airbus (SPOT 6/7)	–	–	F2 @ 0.5–0.95 = 0.82
MASATI-UNet → HRSC2016 (2021)[54, 68]	HRSC (0.41 m RGB)	–	∼0.90*	F1 = 0.945*
UOW-Vessel baseline YOLOv8-L (2024)[56]	UOW-Vessel (HR optical)	–	–	$mAP_{0.5:0.95}$ = 61.7 %
This work (BCE+IoU)	PlanetScope (3.7 m, 6-band)	**0.536**	0.356	Inst. F1 = 0.671

* Exact IoU not reported; value estimated from the authors' confusion matrix.

Table 5.1 shows that the instance-wise IoU performance metrics used for ship detection works. *Instance-wise IoU* refers to the mean IoU per ship instance [69]; *pixel-wise IoU* is the conventional semantic IoU over all pixels [68]. "F-score" denotes the harmonic mean of precision and recall (F1 or variant). **Airbus (2018)** [53]: a Kaggle competition where the metric was an average *F2-score* ($\beta = 2$) over IoU thresholds 0.5–0.95. A basic U-Net achieved F2≈0.82 [11], while winning ensembles reached ∼0.93 (unofficial reports). **MASATI-UNet:** Gallego *et al.'s* MASATI dataset [54] provides labeled

maritime scenes but was used mainly for classification; a U-Net model fine-tuned on HRSC2016 is expected to attain high pixel-level accuracy (no official IoU reported). For context, combining MASATI and HRSC2016 in a detection task yielded F1≈94.5% [70]. **UOW-Vessel (2024):** a new large-scale instance segmentation benchmark [56]. Baseline results show the best model (YOLOv8-L) obtains $mAP \approx 61.7\%$ for instance masks (averaged over IoU=0.5:0.95) and $mAP_{0.5} \approx 80\%$ [56]. This implies many ships are detected with IoU > 0.5, though the overall instance IoU average has room for improvement. In summary, our thesis's primary metric – instance-wise IoU – is not uniformly reported in prior works, but their results (especially the high F2 and mAP scores) indicate that state-of-the-art methods can achieve a high detection/segmentation quality of ships (roughly 80–90% overlap on average) under favorable conditions. The differences in evaluation (instance vs. pixel IoU, F2 vs. mAP) highlight the need to carefully align metrics when comparing models in this domain.

5.3 Error Analysis and Model Limitations

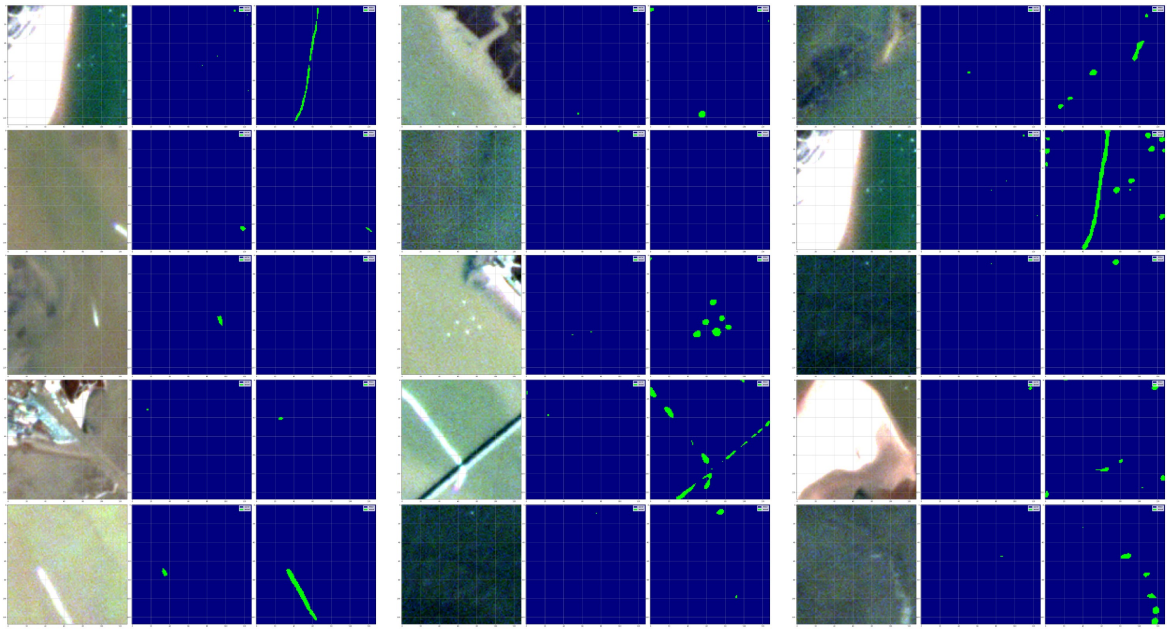
Figure 5.1 shows three typical failure modes of the ship-detection network:

- **False negatives on small or specular targets** (panel a). Tiny vessels or those dominated by a specular highlight are sometimes ignored, yielding an FN and lowering the instance-wise IoU. This behavior is consistent with the instance-level confusion matrix in Fig. 4.5, where 27 of 97 vessels are missed.
- **False positives caused by wakes, coastline texture and thin clouds** (panel b). High-frequency patterns that resemble elongated hulls are incorrectly segmented. Many of these artifacts are removed by the *min_size* = 2 px filter, yet larger blobs survive and reduce precision.
- **Unstable predictions in early epochs** (panel c). During the first ≈30 epochs the model produces scattered, elongated detections; after the BCE + IoU loss "warms up", these artifacts disappear, but slight over-segmentation persists throughout training.

Across all failure cases, predicted masks are ≈10–35% larger than the ground-truth shapes. This systematic bias is expected: with a class imbalance of roughly 1 : 1800 (vessel : water pixels), the BCE + IoU loss rewards over-coverage more than under-coverage. The effect increases recall—desirable for maritime safety—but penalizes the strict instance-IoU metric used in this thesis.

Root causes

- Extreme class imbalance (≈5000 vessel pixels vs. 4.6 M water pixels in the test subset).
- Post-processing thresholds *(IoU thr* = 0.2, *min_size* = 2 px) tuned on BCE-only experiments, not re-optimized for the final BCE + IoU model.
- No geometric or radiometric augmentation during fine-tuning,
- Geographic bias: all training imagery comes from the Western Mediterranean; different coastal morphologies may trigger unseen error modes elsewhere.



(a) BCE+IoU – false negatives and wake over-segmentation

(b) Focal BCE – false positives on wake/coast

(c) Early epoch output – noisy elongated masks

Figure 5.1: Typical failure modes on held-out validation patches. Each triplet displays, from left to right, the RGB input, ground-truth mask, and model prediction.

Mitigation strategies

- Replace the BCE component with a class-balanced or focal variant and re-optimize post-processing thresholds with Bayesian search.
- Introduce synthetic wakes and random rotations during augmentation to reduce orientation bias.
- Expand the training set to Atlantic and Indo-Pacific scenes, then apply lightweight domain adaptation.
- Deploy a confidence-aware alerting pipeline in which detections below an IoU-proxy score of 0.25 are sent to a human operator for validation.

5.4 Summary of Findings

The experimental evidence gathered in Chapter 4 and the error analysis of Section 5.3 lead to four main insights:

1. **Label efficiency.** With only 280 vessel–containing patches the MAE-initialized U-Net reaches *Instance IoU = 0.536* and *Instance F1 = 0.671*, showing that a foundation-model strategy can cut the annotation effort by an order of magnitude compared with fully supervised pipelines.
2. **Loss selection matters.** Adding the IoU term lifts Instance IoU by **+0.155** (0.381 → 0.536) and Instance F1 by **+0.309**, while halving the number of false positives.

3. **Error modes are predictable and actionable.** Systematic over -segmentation, wake confusion and missed micro-boats stem from class imbalance, a fixed matching threshold and limited data diversity – factors that can be mitigated with balanced losses, synthetic augmentation and domain expansion.

4. **Operational viability.** Batch inference processes ∼16 six-band patches every 50 ms on a single NVIDIA A40. Although full-scene timings are still to be profiled, this throughput indicates that end-to-end analysis of PlanetScope strips remains within the near-real-time budget of the MARVISION pipeline.

These findings set the stage for Chapter 6, where we translate the technical results into strategic recommendations for a production-ready maritime-surveillance service.

Chapter 6

Conclusions & Future Work

The final chapter distills the technical results of Chapters 3-5 into a concise set of conclusions and outlines promising directions for extension. It first revisits the objectives stated in Chapter 2, evaluating how far they have been met, and then discusses strategic opportunities for improving accuracy, robustness, and operational integration.

6.1 Conclusions

This work set out to explore whether a foundation-model paradigm—self-supervised pre-training followed by task-specific fine-tuning—could reduce the annotation burden traditionally associated with ship detection in medium-resolution optical imagery. The evidence collected throughout the thesis supports four main conclusions.

1. **Foundation models do pay for themselves.** Pre-training a masked-autoencoder on ∼60 000 unlabeled PlanetScope patches yields an encoder whose features allow the downstream ship-segmentation task to reach Instance *IoU 0.536* and *Instance F1 0.671* with only 280 labeled patches. Compared with a baseline trained from scratch, this represents a gain of +0.155 IoU and +0.309 F1 for roughly one-tenth of the manual labeling effort.

2. **Overlap-aware objectives are critical in extreme class-imbalance scenarios.** The hybrid BCE + IoU loss outperforms focal, Dice, and plain BCE alternatives, halving false positives while keeping recall above 0.64. Directly optimizing the instance overlap metric proves more effective than minimizing average pixel error when vessels occupy <0.1% of the image area.

3. **Qualitative robustness extends beyond numerical metrics.** Multi-spectral inputs (six PlanetScope bands) help the network disambiguate wakes, thin clouds, and pier structures—failure modes prevalent in RGB-only baselines. Although the model tends to over-segment hull boundaries by 10–35%, its predictions remain visually coherent and operationally useful for alerting.

4. **Near-real-time deployment is feasible.** Batch inference processes ≈16 six-band patches per 50 ms on a single NVIDIA A40, implying scene-level runtimes compatible with the MARVISION ingestion rate. The 61 M-parameter network (≈245 MB ONNX) fits comfortably within the memory budget of the target processing node.

Collectively, these findings validate the central hypothesis: **a self-supervised foundation model can deliver accurate, label-efficient, and operationally viable ship detection on PlanetScope imagery**, paving the way for scalable maritime-surveillance services.

6.2 Future Work

Building on the strengths and limitations identified in Chapter 5, six priority lines of work are proposed. Each addresses a concrete limitation of the current prototype while remaining realistic within the MARVISION roadmap.

1. **Domain expansion to new seas**
   Integrate PlanetScope imagery from the Atlantic and Indo-Pacific, followed by light fine–tuning (20-30 new scenes). This will quantify geographic generalization and reduce the Western-Mediterranean bias seen in Section 5.3.

2. **Synthetic wakes and rotation augmentations**
   Extend the augmentation pipeline with procedural wake overlays, random rotations and specular noise. The goal is to cut false positives on wakes/coastlines and improve recall on oblique vessels.

3. **Balanced or focal IoU loss**
   Replace the BCE term with a class-balanced focal component to penalize over-segmentation and boost minority pixels, then re-evaluate instance IoU on the current test split.

4. **Multi-resolution fusion (PlanetScope + Sentinel-2)**
   Develop a dual–branch encoder that ingests co–registered PlanetScope ($\sim$3 m GSD) and Sentinel–2 MSI (10 m GSD) patches. The goal is to combine PlanetScope's fine spatial detail with Sentinel-2's broader context, yielding scale–robust detection while remaining within the optical domain. This step is facilitated by the six-band PlanetScope subset already employed in this thesis, which was selected for interoperability with the corresponding Sentinel-2 bands.

5. **Continual learning pipeline**
   Deploy an automated monthly fine-tuning loop that ingests new analyst labels, using rehearsal buffers or elastic weight consolidation to avoid catastrophic forgetting.

6. **Model compression and ONNX quantization**
   Quantize the 61 M-parameter network to INT8 and prune redundant channels, targeting a footprint <90 MB and real-time inference on edge hardware or future on-board processing units.

Bibliography

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241, Springer, 2015. ix, 22, 23

[2] V. W. G. H, *Satellite Imaging for Maritime Surveillance of the European Seas*. Berlin (Germany): Springer Science+Business Media B.V., 2008. 1

[3] A. Xiao, W. Xuan, J. Wang, J. Huang, D. Tao, S. Lu, and N. Yokoya, "Foundation models for remote sensing and earth observation: A survey," *arXiv preprint arXiv:2410.16602*, 2024. 1

[4] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021. 1

[5] S. Lu, J. Guo, J. R. Zimmer-Dauphinee, J. M. Nieusma, X. Wang, S. A. Wernke, Y. Huo, *et al.*, "Vision foundation models in remote sensing: A survey," *IEEE Geoscience and Remote Sensing Magazine*, 2025. 1

[6] S. D. Priyadharashini and K. Vadivazhagan, "Enhanced vessel detection for maritime surveillance using hyperparameter-tuned deep learning on sar images," *South Eastern European Journal of Public Health*, 2025. 5

[7] A. A. Adegun, J. V. Fonou Dombeu, S. Viriri, and J. Odindi, "State-of-the-art deep learning methods for objects detection in remote sensing satellite images," *Sensors*, 2023. 5, 6, 9, 10, 12, 13, 14

[8] S. Lu, J. Guo, J. R. Zimmer-Dauphinee, J. M. Nieusma, X. Wang, P. VanValkenburgh, S. A. Wernke, and Y. Huo, "Vision Foundation Models in Remote Sensing: A Survey," *arXiv e-prints*, 2024. 5, 8, 10, 12, 17

[9] N. Li, X. Pan, L. Yang, Z. Huang, Z. Wu, and G. Zheng, "Adaptive cfar method for sar ship detection using intensity and texture feature fusion attention contrast mechanism," *Sensors*, 2022. 6, 7, 10, 11, 12

[10] X. Qin, S. Zhou, H. Zou, and G. Gao, "A cfar detection algorithm for generalized gamma distributed background in high-resolution sar images," *IEEE Geoscience and Remote Sensing Letters*, 2012. 6

[11] K. Patel, C. Bhatt, and P. L. Mazzeo, "Deep learning-based automatic detection of ships: An experimental study using satellite images," *Journal of imaging*, 2022. 7, 9, 13, 44

[12] Z. Chen, C. Liu, V. F. Filaretov, and D. A. Yukhimets, "Multi-scale ship detection algorithm based on yolov7 for complex scene sar images," *Remote Sensing*, 2023. 7

[13] A. Nambiar, A. Vaigandla, and S. Rajendran, "Efficient ship detection in synthetic aperture radar images and lateral images using deep learning techniques," in *OCEANS 2022, Hampton Roads*, 2022. 7

[14] Y. Zhang, M. J. Er, W. Gao, and J. Wu, "High performance ship detection via transformer and feature distillation," in *2022 5th International Conference on Intelligent Autonomous Systems (ICoIAS)*, 2022. 7, 10

[15] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017. 7

[16] Z. Zhang, X. Qiu, and Y. Li, "Sefpn: Scale-equalizing feature pyramid network for object detection," *Sensors*, 2021. 7

[17] C. Picron and T. Tuytelaars, "Trident pyramid networks for object detection," *Proceedings BMVC 2022*, 2022. 7

[18] Y. Gan, S. You, Z. Luo, K. Liu, T. Zhang, and L. Du, "Object detection in remote sensing images with mask r-cnn," in *Journal of Physics: Conference Series*, 2020. 7

[19] J. Liu, D. Yang, and F. Hu, "Multiscale object detection in remote sensing images combined with multi-receptive-field features and relation-connected attention," *Remote Sensing*, 2022. 7

[20] F. Gao, Y. He, J. Wang, A. Hussain, and H. Zhou, "Anchor-free convolutional network with dense attention feature aggregation for ship detection in sar images," *Remote Sensing*, 2020. 7, 14

[21] T. Zhang, X. Zhang, J. Li, X. Xu, B. Wang, X. Zhan, Y. Xu, X. Ke, T. Zeng, H. Su, I. Ahmad, D. Pan, C. Liu, Y. Zhou, J. Shi, and S. Wei, "Sar ship detection dataset (ssdd): Official release and comprehensive data analysis," *Remote Sensing*, 2021. 7, 8, 13, 15

[22] J. Li, C. Qu, and J. Shao, "Ship detection in sar images based on an improved faster r-cnn," in *2017 SAR in Big Data Era: Models, Methods and Applications (BIGSARDATA)*, 2017. 7, 8, 13

[23] Y. Yang, J. Chen, L. Sun, Z. Zhou, Z. Huang, and B. Wu, "Unsupervised domain-adaptive sar ship detection based on cross-domain feature interaction and data contribution balance," *Remote Sensing*, 2024. 7

[24] H. Dai, L. Du, Y. Wang, and Z. Wang, "A modified cfar algorithm based on object proposals for ship target detection in sar images," *IEEE Geoscience and Remote Sensing Letters*, 2016. 7

[25] F. M. Vieira, F. Vincent, J.-Y. Tourneret, D. Bonacci, M. Spigai, M. Ansart, and J. Richard, "Ship detection using sar and ais raw data for maritime surveillance," in *2016 24th European Signal Processing Conference (EUSIPCO)*, 2016. 7

[26] Z. Yuan, Y. Li, Y. Liu, J. Liang, and Y. Zhang, "Unsupervised ship detection in sar imagery based on energy density-induced clustering," *International Journal of Network Dynamics and Intelligence*, 2023. 8, 10

[27] O. Manas, A. Lacoste, X. Giró-i-Nieto, D. Vazquez, and P. Rodriguez, "Seasonal contrast: Unsupervised pre-training from uncurated remote sensing data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 8

[28] J. Prexl and M. Schmitt, "Multi-modal multi-objective contrastive learning for sentinel-1/2 imagery," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2023. 8

[29] X. Wanyan, S. Seneviratne, S. Shen, and M. Kirley, "Extending global-local view alignment for self-supervised learning with remote sensing imagery," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 8

[30] Y. Cong, S. Khanna, C. Meng, P. Liu, E. Rozi, Y. He, M. Burke, D. Lobell, and S. Ermon, "Satmae: Pre-training transformers for temporal and multi-spectral satellite imagery," *Advances in Neural Information Processing Systems*, 2022. 8

[31] C. J. Reed, R. Gupta, S. Li, S. Brockman, C. Funk, B. Clipp, K. Keutzer, S. Candido, M. Uyttendaele, and T. Darrell, "Scale-mae: A scale-aware masked autoencoder for multiscale geospatial representation learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 8

[32] U. Mall, B. Hariharan, and K. Bala, "Change-aware sampling and contrastive learning for satellite images," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 8

[33] U. Mall, C. P. Phoo, M. K. Liu, C. Vondrick, B. Hariharan, and K. Bala, "Remote sensing vision-language foundation models without annotations via ground remote alignment," *arXiv preprint arXiv:2312.06960*, 2023. 8, 17

[34] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, 2021. 8

[35] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023. 8

[36] L. Pang, B. Li, F. Zhang, X. Meng, and L. Zhang, "A lightweight yolov5-mne algorithm for sar ship detection," *Sensors*, 2022. 9

[37] O. E. Olorunshola, M. E. Irhebhude, and A. E. Evwiekpaefe, "A comparative study of yolov5 and yolov7 object detection algorithms," *Journal of Computing and Social Informatics*, 2023. 9

[38] M. Li, Z. Zhang, L. Lei, X. Wang, and X. Guo, "Agricultural greenhouses detection in high-resolution satellite images based on convolutional neural networks: Comparison of faster r-cnn, yolo v3 and ssd," *Sensors*, 2020. 9

[39] J. Deng, X. Xuan, W. Wang, Z. Li, H. Yao, and Z. Wang, "A review of research on object detection based on deep learning," in *Journal of Physics: Conference Series*, 2020. 9

[40] K. Tong, Y. Wu, and F. Zhou, "Recent advances in small object detection based on deep learning: A review," *Image and Vision Computing*, 2020. 10

[41] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, 2014. 10, 14

[42] L. Lang, K. Xu, Q. Zhang, and D. Wang, "Fast and accurate object detection in remote sensing images based on lightweight deep neural network," *Sensors*, 2021. 10

[43] D. Heller, M. Rizk, R. Douguet, A. Baghdadi, and J.-P. Diguet, "Marine objects detection using deep learning on embedded edge devices," in *2022 IEEE International Workshop on Rapid System Prototyping (RSP)*, 2022. 10

[44] Y. Li, H. Yuan, Y. Wang, and B. Zhang, "Maritime vessel detection and tracking under uav vision," in *2022 International Conference on Artificial Intelligence and Computer Information Technology (AICIT)*, 2022. 10

[45] S. Cheng, Z. Yishuang, and S. Wu, "Deep learning based efficient ship detection from drone-captured images for maritime surveillance," *Ocean Engineering*, 2023. 10

[46] P. Jajal, W. Jiang, A. Tewari, E. Kocinare, J. Woo, A. Sarraf, Y.-H. Lu, G. K. Thiruvathukal, and J. C. Davis, "Analysis of failures and risks in deep learning model converters: A case study in the onnx ecosystem," *arXiv preprint arXiv:2303.17708*, 2023. 11

[47] P. Xu, Q. Li, B. Zhang, F. Wu, K. Zhao, X. Du, C. Yang, and R. Zhong, "On-board real-time ship detection in hisea-1 sar images based on cfar and lightweight deep learning," *Remote Sensing*, 2021. 11

[48] S. Wei, X. Zeng, Q. Qu, M. Wang, H. Su, and J. Shi, "Hrsid: A high-resolution sar images dataset for ship detection and instance segmentation," *IEEE Access*, 2020. 13

[49] T. Zhang, X. Zhang, X. Ke, X. Zhan, J. Shi, S. Wei, D. Pan, J. Li, H. Su, Y. Zhou, and D. Kumar, "Ls-ssdd-v1.0: A deep learning dataset dedicated to small ship detection from large-scale sentinel-1 sar images," *Remote Sensing*, 2020. 13

[50] S. Xian, W. Zhirui, S. Yuanrui, D. Wenhui, Z. Yue, and F. Kun, "Air-sarship-1.0: High-resolution sar ship detection dataset," *Journal of radars*, 2019. 13

[51] Z. Wang, K. Yuzhuo, Z. Xuan, W. Yuelei, Z. Ting, and S. Xian, "Sar-aircraft-1.0: High-resolution sar aircraft detection and recognition dataset," *Journal of radars*, 2023. 13

[52] R. Hammell, "Ships in satellite imagery," 2018. 13

[53] inversion, J. Faudi, and Martin, "Airbus ship detection challenge," 2018. 13, 15, 44

[54] A.-J. Gallego, A. Pertusa, and P. Gil, "Automatic ship classification from optical aerial images with convolutional neural networks," *Remote Sensing*, 2018. 13, 44

[55] S. Alashhab, A.-J. Gallego, A. Pertusa, and P. Gil, "Precise ship location with cnn filter selection from optical aerial images," *IEEE Access*, 2019. 13

[56] L. Bui, S. L. Phung, Y. Di, H. T. Le, N. T. T. Phong, B. Sandy, and A. Bouzerdoum, "Uow-vessel: A benchmark dataset of high-resolution optical satellite images for vessel detection and segmentation," *Proceedings - 2024 IEEE Winter Conference on Applications of Computer Vision*, 2024. 13, 15, 44, 45

[57] Z. Zhang, L. Zhang, Y. Wang, P. Feng, and R. He, "Shiprsimagenet: A large-scale fine-grained dataset for ship detection in high-resolution optical remote sensing images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2021. 13

[58] N. Sergievskiy and A. Ponamarev, "Reduced focal loss: 1st place solution to xview object detection in satellite imagery," *arXiv preprint arXiv:1903.01347*, 2019. 14

[59] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, "Dota: A large-scale dataset for object detection in aerial images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018. 14

[60] M. Everingham, S. M. Eslami, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *Int. J. Comput. Vision*, 2015. 14

[61] Sentinel Hub, "PlanetScope." https://docs.sentinel-hub.com/api/latest/data/planet/planet-scope/, 2024. Accessed: 2024-04-04. 19

[62] P. L. PBC, "Planet application program interface: In space for life on earth," 2025. 19

[63] S. Gillies *et al.*, "Rasterio: geospatial raster i/o for Python programmers," 2013–. 21, 22

[64] C. Fibaek, L. Camilleri, A. Luyts, N. Dionelis, and B. Le Saux, "Phileo bench: Evaluating geo-spatial foundation models," in *IGARSS 2024-2024 IEEE International Geoscience and Remote Sensing Symposium*, pp. 2739–2744, IEEE, 2024. 22

[65] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019. 23

[66] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16000–16009, June 2022. 25

[67] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 658–666, 2019. 25, 26

[68] Z. Liu, L. Yuan, L. Weng, and Y. Yang, "A high resolution optical satellite image dataset for ship recognition and some new baselines," in *Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods - ICPRAM*, pp. 324–331, INSTICC, SciTePress, 2017. 44

[69] D. Wang, J. Zhang, B. Du, M. Xu, L. Liu, D. Tao, and L. Zhang, "Samrs: Scaling-up remote sensing segmentation dataset with segment anything model," *Advances in Neural Information Processing Systems*, vol. 36, pp. 8815–8827, 2023. 44

[70] P. Antunes and A. Podobas, "Fpga-based neural network accelerators for space applications: A survey," *arXiv preprint arXiv:2504.16173*, 2025. 45