WAVETRACK.AI

SHIP DETECTION USING SAR IMAGES

Ángel Cuadrado Serrano • Aman Arora •
Gabriela Nutsova • Ivan Kanev • Nikola Lilov

Institute:	Space Challenges 2025
Team:	WaveTrack.AI
Team Members:	Ángel Cuadrado Serrano, Aman Arora, Gabriela Nutsova, Ivan Kanev, Nikola D. Lilov
Title:	Ship Detection Using SAR Images
Abstract:	Maritime domain awareness is more critical than ever due to increasing geopolitical tensions, but traditional surveillance methods have major limitations and high costs. This project introduces a new, cost-effective solution that overcomes these challenges. While traditional optical methods are limited by weather and lighting conditions, our approach leverages the all-weather, day-and-night capabilities of radar. We have developed a containerized AI model that operates directly on a satellite, enabling onboard processing of radar imagery. Instead of downlinking huge image files for ground-based analysis, the AI autonomously detects ships and extracts only essential information, compiling it into a small, lightweight JSON file. This approach drastically reduces the cost and time required for data transfer and subsequent analysis, making continuous maritime surveillance efficient and affordable. Leveraging the YOLOv5n deep learning model, our system transforms a premium capability into an accessible tool for nations regardless of their economic standing. Looking ahead, our platform is planned to expand its capabilities to not only detect ships but also estimate their velocity. This work contributes a scalable solution that enhances maritime situational awareness, supporting national security.
Supervisor:	Krasimir Stoev
Date:	18/08/2025

Table of Content

1 Introduction

1.1 Motivation

The increasing geopolitical tensions make effective maritime surveillance essential. The need for real time and accurate ship detection is central. However, traditional surveillance methods have major limitations and high costs.

Visual observation and patrol boats have a limit on the visible range and are weather dependent. The tracking technology used on ships to broadcast their location, while useful, can be easily turned off or manipulated. Optical satellite imagery is hindered by cloud cover and darkness.

Radar imagery offers a significant solution, as it can penetrate clouds and operate day or night, providing a consistent, all-weather view. Unfortunately, the high cost of collecting and downlinking vast amounts of raw data is a significant barrier.

Our project provides a solution to these challenges by accelerating both data transfer and analysis. We have developed a containerised AI model that operates directly on a satellite, enabling onboard processing of radar imagery. This approach eliminates the need to downlink massive image files for ground-based analysis. Instead, the AI model autonomously detects ships in orbit, and extracts only the essential information which is then compiled into a small, lightweight JSON file. By transmitting only this highly compressed data, we drastically reduce both the cost and time required for data transfer and subsequent detection.

This product is striving to make continuous maritime surveillance efficient and affordable, transforming it from a premium capability into a cost effective tool, easily accessible by states, regardless of their economic stance.

1.2 Project Scope and Objectives

The primary objective of this project is to design and implement a lightweight, high-performance AI model for ship detection from SAR imagery, with focus on real-world deployment aboard an operational satellite. The model will be optimised to operate under the strict computational and memory constraints of space borne systems, ensuring efficient inference without sacrificing detection accuracy.

- Ensure the model can be trained effectively on diverse SAR datasets of varying resolutions, noise characteristics, and sensor sources, while maintaining strong generalisation performance across unseen operational scenarios.
- Take advantage of SAR's unique ability to penetrate clouds and operate independently of

daylight, enabling continuous maritime surveillance regardless of weather conditions or time of day.

- Design the model and its preprocessing pipeline to integrate seamlessly into satellite-based workflows, enabling near-real-time ship detection directly in orbit and minimizing downlink bandwidth requirements.
- Detect vessels across a wide range of environments, from open ocean to high-clutter coastal and port regions, supporting enhanced maritime situational awareness.

By achieving these objectives, the project aims to deliver a deployable AI solution that provides timely, reliable, and scalable ship detection from SAR imagery, supporting operational decision-making in both civilian and defence maritime domains.

2 Work Breakdown and Team Structure

The project is divided into five major work packages, each addressing a critical component of the system:

WP1 Data Acquisition and Preparation: Understanding how SAR data is saved, processed, and visualised. Understanding how Sentinel-1 operates. Researching open source dataset (HRSID), and annotation quality checks.

WP2 Preprocessing and Denoising: Implementation of the preprocessing pipeline to improve the quality of SAR imagery for ship detection. This includes speckle noise reduction using FABF, followed by contrast enhancement. Preserves edges while smoothing uniform regions from speckles.

WP3 AI Model Development and Evaluation: Training and validation of YOLOv5n for ship detection, implementation of denoising, data augmentation strategies and end-to-end pipeline, and benchmarking accuracy and inference performance on Jetson AGX Orin.

WP4 System Integration and Deployment: Model evaluations, and integration of the product with web backend/frontend for interactive inference demo.

WP5 Research, Documentation, and Future Directions: Literature review, background study of SAR imaging, analysis of current state of the art, report writing, and investigation of extensions such as ship velocity estimation.

Team Structure and Roles The team consisted of five members, each contributing according to their expertise:

1. Team Leader (Gabriela Nutsova): Coordinated project activities, managed timelines, and facilitated communication. Contributed to SAR data acquisition tests from Copernicus and prepared supporting documentation.
2. AI and Software Lead (Aman Arora): Designed and implemented the core AI pipeline, trained YOLOv5n models, optimized inference for Jetson AGX Orin, and integrated the Dockerized software stack.
3. Lead Researcher and Physicist (Ivan Kanev): Conducted literature review and background research on SAR, authored major sections of the report, dataset analysis, and investigated methods for ship velocity estimation.
4. AI Evaluation Support and Web Developer (Ángel Cuadrado Serrano): Assisted in training runs, contributed to model evaluation and hyperparameter tuning, and supported improvements to the detection pipeline. Developed the PoC demo web-app.
5. Image Processing Lead (Nikola D. Lilov): Designed and implemented the preprocessing pipeline, utilizing GPU capabilities like those on the Jetson AGX Orin, facilitated the dataset acquisition, and contributed to overall methodology and product design.

This structure ensured coverage of all aspects of the project: data, preprocessing, AI development, deployment, and research documentation, with clear ownership of responsibilities.

3 Background and State of the Art

Maritime domain awareness has become an essential priority for governments, coast guards, and private enterprises. Whether for tracking fishing activity, identifying illegal shipping, or responding to emergencies, the ability to reliably detect vessels in vast ocean regions is critical. Traditional optical imaging solutions face serious limitations due to weather, cloud cover, and lighting conditions. Synthetic Aperture Radar (SAR), with its ability to operate in all weather and lighting conditions, provides a powerful alternative for persistent maritime surveillance.

This project focuses on the development of an AI-based system for automatic detection of ships in SAR imagery. By leveraging deep learning models trained on labelled SAR datasets, this project aims to significantly enhance detection accuracy and processing speed compared to conventional methods.

3.0.1 Sentinel-1

Sentinel-1 (S-1) is a radar imaging satellite mission developed by the European Space Agency (ESA) under the Copernicus Program. It provides all-weather, day-and-night Earth observation capabilities, with a particular focus on monitoring land and ocean surfaces. In April 2014 Sentinel-1A (S-1A) was launched, followed by Sentinel-1B (S-1B) in April 2016. In recent history Sentinel-1C (S-1C), launched in 2024, and Sentinel-1D (S-1D), expected launch in 2025. S-1B was decommissioned in 2022 due to unresolvable power supply issues. The S-1 satellites operate in a sun-synchronous polar orbit at 693 km altitude. The ground track repeat cycle is 12 days per satellite, 6 days when both S-1A and S-1B were operational.

The instrumentation on board S-1A is an active C-band Synthetic Aperture Radar (SAR), operating at ~ 5.405 GHz (wavelength ~ 5.6 cm). The key features of the satellite are: 1) active radar, which emits microwave pulses and receives backscattered signals day and night, 2) penetrates clouds, rain, and darkness, 3) delivers high-resolution imagery, up to 5 meters. (sen, 2025a)

The satellite constellation has four modes (sen, 2025a):

a. Strip Map mode (SM): This mode works by illuminating a specific strip of land, while the satellite is always looking at the same relative location on the ground while the platform moves forward. It is typically used to monitor small islands and emergency management for weather events. This method can be seen in figure 1 (Strip map Mode).

b. Extra Wide Swath (EW): EW mode also uses TOPSAR. The difference is that it implements smaller swaths. This mode is used over sea ice, polar zones and certain maritime areas. It aids the research in ice physics, oil spill monitoring and security services. Furthermore, EW mode can also be used for interferometry as it shares the same characteristics for synchronisation of burst, baseline, and Doppler stability. This method is illustrated in figure 1 (Extra Wide Swath Mode)

c. Wave: In this mode data is acquired in small strip map scenes called 'vignettes'. The vignettes are acquired by alternating observing one vignette at a near range incidence angle while the next vignette is observed at a far range incidence angle. Wave mode is the main operational mode over open sea, with data only available in single polarization. This method is illustrated in figure 1 (Wave Mode)

3.0.2 Synthetic Aperture Radar

SAR is an advanced form of radar technology used to create high-resolution images of the Earth's surface. Unlike optical imaging systems, SAR is an active remote sensing instrument, meaning it generates its own microwave signals and measures the energy that is scattered back to the sensor.

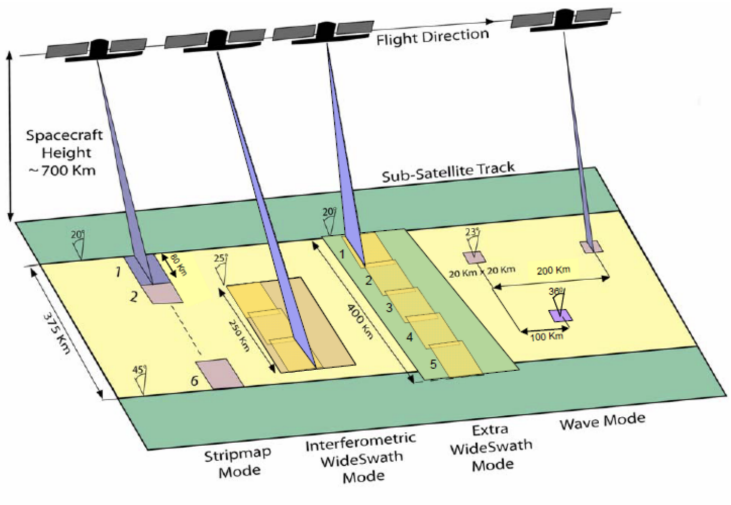
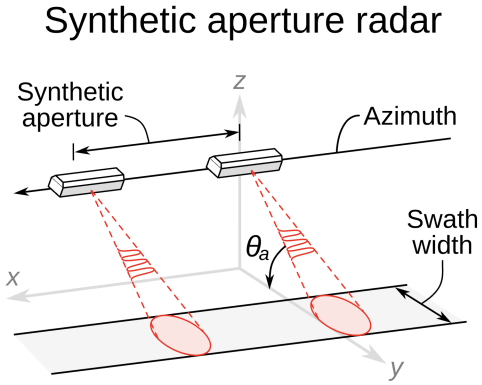


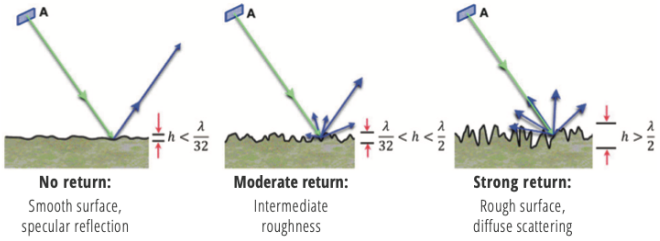
Figure 1: S-1 Acquisitions Swath Coverage (sen, 2025a)

The fundamental principle behind SAR is the use of the motion of the radar platform, which is typically mounted on a satellite, like S-1, or an aircraft, to simulate a very large antenna.

As the satellite platform moves along its orbit, the SAR instrument transmits microwave pulses toward the Earth's surface and records the backscattered signals that return. By precisely measuring the time delay and phase of the returned echoes and processing them coherently, SAR systems are able to synthesize a large virtual antenna, hence the name "synthetic aperture", and achieve much higher spatial resolution than would be possible with a physically small antenna. (ear, 2025). This schematic is visualised in figure 2 (a).



(a) Illustration of Synthetic Aperture Radar (Wikipedia contributors, 2025)



(b) Backscatter return (sen, 2025b)

Figure 2: Illustration of the concept behind a Synthetic Aperture Radar

The strength of the returned signal is quantified using the radar backscatter coefficient, denoted σ^0, which describes the fraction of the radar energy that is scattered back to the sensor per unit area. Illustration of this can be seen in figure 2b. This coefficient is a key metric in SAR imagery

and depends on surface properties such as roughness, moisture content, and geometry. σ^0 can be expressed in two ways: in linear scale, which is a unitless ratio (m^2/m^2), and in logarithmic scale, expressed in decibels (dB) as

$$dB_0 = 10 \cdot log_{10}(\sigma^0) \tag{1}$$

The log scale is more common for visualization purposes. In this scale, low-reflectivity surfaces such as calm water may appear around –30 dB, while highly reflective surfaces like urban areas or rough terrain can reach 0 dB or higher. This project will use linear scale, as that is the most suitable scale for training a machine learning model.

3.1 State of the art analysis

Having introduced S-1 and the principles of SAR, this section will discuss the state of the art in C-band SAR Earth observation, covering the choices of hardware, data, and software, that were made for this project.

3.1.1 Choice of hardware

This project uses Sentinel-1 to acquire SAR images. The reasoning behind this choice will be listed in this section. Table 1 lists a handful of SAR satellites, which are currently in orbit. These satellites have different bands, modes, and polarisation, specifically chosen for each mission. The different uses for each band are listed here:

- X band: High resolution. Sensitive to surface roughness, with limited penetration. Ideal for surface overview in a region, such as urban mapping and infrastructure.
- C band: Balanced resolution vs penetration. Global standard for Earth Observation. Ideal for general observations, but with extra details
- S band: Intermediate penetration. Useful for vegetation and soil moisture monitoring.
- L band: Deep penetration into medium. Ideal for forestry, biomass, tectonic deformation, landslides

Mission	Band	Key Modes	Polarisations
Sentinel-1	C (5.4 GHz)	SM, IW, EW, WV (TOPS)	Single / Dual
RADARSAT	C	Spotlight, High / Med / Low Res, ScanSAR, Ship Detection	Single / Dual / Quad / Compact
RADARSAT-2	C	Spotlight, Ultra-Fine, Fine, ScanSAR	Single / Dual / Quad
TerraSAR-X	X (9.65 GHz)	Staring Spot, HR Spot, Spot, StripMap, ScanSAR	Single / Dual
PAZ	X	Staring Spot, HR Spot, Spot, StripMap, ScanSAR	Single / Dual
ALOS-2	L (1.2–1.3 GHz)	Spotlight, Ultra-Fine/High-Sens., Fine, ScanSAR	Multiple
SAOCOM-1A/1B	L	Stripmap; TOPSAR Narrow/Wide	Multiple
NovaSAR-1	S (3.1–3.3 GHz)	Stripmap, ScanSAR, Maritime (AIS synergy)	Single / Dual / Tri

Table 1: Comparison of SAR satellites in orbit

This project requires data with high temporal resolution, open accessibility, and ease of use. From Table 1, this narrows the options to Sentinel-1, SAOCOM-1A/1B, and NovaSAR-1. The project's band requirements lie in a sweet spot between spatial detail and penetration depth, which further emphasizes the suitability of C-band. Among these missions, only Sentinel-1 fully meets the criteria. Although missions like RADARSAT-2 and TerraSAR-X offer higher resolution and more polarization flexibility, their commercial access severely restricts scientific and operational uptake. By contrast, Sentinel-1's free and systematic acquisitions have made it the de facto global standard in C-band SAR Earth observation. Moreover, Sentinel-1 was specifically designed to be agile in terms of imaging modes and polarisation, enabling configuration of acquisition settings to optimize information extraction.

This project uses Sentinel-1 data acquired in dual polarisation (VV + VH) and in Interferometric Wide Swath (IW) mode. The choice of dual polarisation is motivated by the fact that combining co-polarized (VV) and cross-polarized (VH) channels provides a more comprehensive representation of sea surface conditions. VV polarization primarily captures surface scattering, while VH is sensitive to volume. Regarding imaging mode, IW was selected because it is the Sentinel-1 standard mode for global Earth observation, providing consistent coverage over both land and sea. This is important, as some of the images will be of coastal areas.

3.1.2 Choice of data

Ship detection using SAR imagery has been an active research field for several decades. In the early stages, progress was hindered by the lack of standardized, high-quality SAR datasets, making reliable detection difficult. With the advent of AI in this field, however, the need for large-scale, high-resolution datasets became increasingly critical. This project therefore reviewed and compared the leading publicly available datasets. The five most relevant datasets considered are summarized in Table 2.

Dataset	**Task**
SAR Ship Detection Dataset	Detection & Semantic Segmentation
OpenSARship	Detection
High-Resolution SAR Images Dataset	Detection & Instance Segmentation
SAR Ship Detection Dataset	Detection & Instance Segmentation
xView3-SAR	Detection & Classification
SARFish	Detection & Single Look Complex data

Table 2: Comparison of open source SAR datasets

To ensure the dataset is well-suited for training a robust and practical ship detection model, several key requirements should be considered:

a. Sensor Consistency: Focusing solely on S-1 SAR imagery ensures uniform sensor characteristics (noise profile, resolution), simplifying model training and reducing domain adaptation overhead. In a real-world scenario, the model would also be trained only on one sensor (the satellite), so this is reasonable.

b. Annotation Quality: Ship positions must be well-labelled, ideally with bounding boxes.

c. Environmental Diversity: The dataset ought to include data that captures the diversity and complexity of real-world maritime environments, ensuring the model can generalize well across different scenarios:

   - Images without any ships
   - Open-ocean vessels (for baseline detection accuracy)
   - Ships near ports, coastlines, and especially dockyards, to ensure robustness in cluttered or ambiguous scenes

Based on the requirements stated above the best suited dataset for this project's goals is the High-Resolution SAR Images Dataset (HRSID) Wei et al. (2020).

3.1.3 Choice of Software

A review of prior work in SAR ship detection revealed diverse methodological approaches ranging from classical techniques such as CFAR and GLRT-based models on Sentinel-1 data (Iervolino et al., 2015), to early deep learning models leveraging contextual features (Hu et al., 2025), (Grover et al., 2018). Lightweight and on-board deployable architectures, such as Lite-YOLOv5 (Xu et al., 2022), have been explored for real-time applications on large-scene SAR images. Hybrid frameworks integrating multi-frequency SAR data (Del Prete et al., 2023) and sensor fusion with AIS (Achiri et al., 2018) have further advanced detection robustness. Recent works have also addressed computational efficiency for embedded deployment (Chen et al., 2023a), supporting the development of our custom architecture. This survey informed the design decisions behind the architecture employed in our project.

The ship detection model implemented in this system is based on the You Only Look Once (YOLO) family of real-time object detectors, specifically the lightweight YOLOv5n variant. This choice followed a comparison with alternative architectures such as U-Net (optimized for dense segmentation), Faster R-CNN (high accuracy but higher latency), and Mask R-CNN (instance segmentation). YOLOv5n offered the best balance between detection accuracy, inference speed, and resource efficiency for deployment on the required NVIDIA Jetson AGX Orin platform. The architecture diagram of YOLOv5 can be seen in Figure 3.

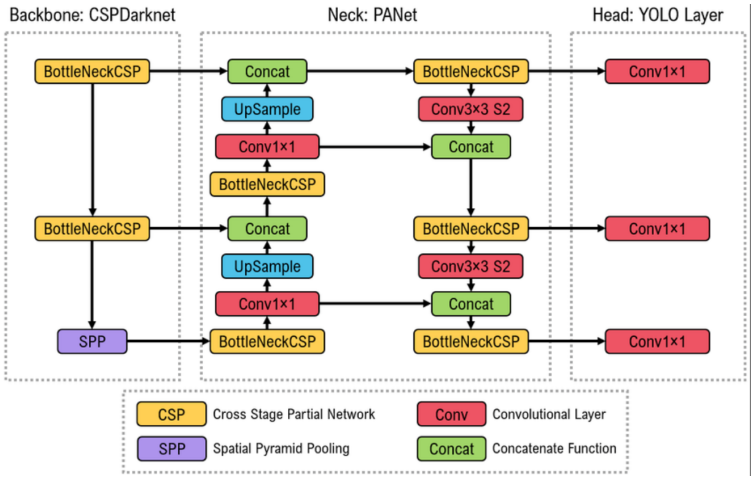


Figure 3: YOLOv5 architecture (Katsamenis et al., 2022)

The main reasons for selecting YOLOv5n are:

- Single-Stage Architecture: Performs classification and bounding box regression in a single pass, enabling low-latency inference suitable for real-time or near-real-time applications.
- High Efficiency: Compact model size and reduced parameter count make it feasible for embedded deployment, operating efficiently within the Jetson AGX Orin's 8–16 GB memory

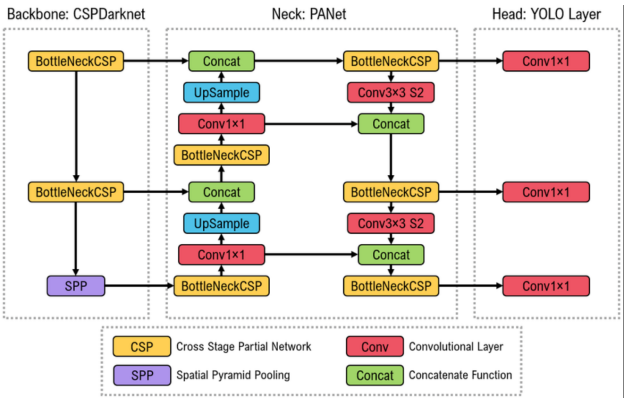


Figure 4: Enter Caption

modes.

- SAR Adaptability: The anchor-based prediction pipeline generalizes well to the wide range of ship scales and aspect ratios in Sentinel-1 SAR imagery.
- Community Support: YOLOv5 has extensive open-source adoption, ensuring availability of pretrained weights, documentation, and extensions.

During the early stages of the project, it was experimented with an enhanced variant we referred to in our implementations as "SAR-YOLO," which incorporated attention mechanisms and modified feature pyramids (Chen et al., 2023b; Hou et al., 2021; Zhang et al., 2023). While this model provided comparable accuracy, it was significantly heavier in terms of inference speed and memory usage. Based on empirical evaluation, YOLOv5n was ultimately adopted as the final model, since it achieved nearly identical accuracy ($mAP@0.5 \approx 0.91$) while running substantially faster and with a smaller memory footprint. This trade-off was critical for ensuring reliable deployment on the Jetson AGX Orin under real-world, resource-limited conditions.

4 System Requirements

4.1 Functional Requirements

FR1. Ship Detection: The system shall detect vessels in Sentinel-1 SAR images with configurable confidence and IoU thresholds (default: 0.5 and 0.45).

FR2. Image Preprocessing: The system shall apply denoising (fast adaptive bilateral filtering) prior to inferencing to reduce noise and false positives.

FR3. Data Augmentation: The training pipeline shall support augmentations including rotation, scaling, and noise simulation to improve robustness.

FR4. Output Generation: The system shall output detection results as bounding boxes with confidence scores, both in YOLO-format labels and annotated images.

FR5. Packaging: The system shall generate lightweight detection summaries (JSON/JSONL, thumbnails) suitable for transmission or user review.

4.2 Performance Requirements

PR1. Detection Accuracy: The system shall achieve a minimum mAP@0.5 of 0.75 on the HRSID validation set.

PR2. Precision/Recall: The system shall maintain at least 0.80 precision and 0.70 recall under default thresholds.

PR3. Inference Speed: The system shall process a 640×640 image in under 0.5 seconds on GPU (Jetson Xavier NX) and under 2 seconds on CPU.

PR4. Resource Usage: The system shall operate within 8 GB RAM to support embedded deployment.

4.3 Technical Requirements

TR1. Frameworks: The system shall be implemented in Python (3.8) using PyTorch, OpenCV, and NumPy.

TR2. Model Architecture: The system shall be based on YOLOv5 (nano variant) with SAR-specific preprocessing.

TR3. Containerization: The system shall support Docker-based deployment on both x86_64 (development) and ARM64 (Jetson) platforms.

TR4. Model Conversion: The system shall support export to ONNX and TensorRT for optimized inference.

5 Contributions

The project makes the following contributions:

- Curated SAR datasets for ship detection: Trained and evaluated on HRSID (5,604 images; 16,951 ship instances) with 640×640 letterbox inputs, and created specialized HRSID_land and HRSID_augmented variants for near-land ship detection scenarios.

- SAR-specific preprocessing pipeline: Implemented Fast Adaptive Bilateral Filtering (FABF) for speckle denoising while preserving ship edges, with configurable parameters (rho, N, sigma) and integration into the YOLO training and inference pipeline.
- Enhanced YOLOv5-based detector: Extended standard YOLOv5 with custom RFA (Receptive Field Attention) modules including RFCBAMConv and RFCAConv for improved feature extraction, while maintaining compatibility with the original YOLO architecture and training pipeline.
- Comprehensive evaluation framework: Achieved mAP@0.5 of 0.914 on HRSID validation set with YOLOv5n, providing detailed metrics including precision, recall, F1-score, and confusion matrix analysis for both training and validation phases.
- Production-ready deployment system: Implemented a complete Docker-based pipeline (currently working for x86_64 architecture, and work in progress for NVIDIA Jetson AGX Orin deployment) support for satellite deployment, including ingest services, georeferencing, post-processing, and health monitoring, alongside a command-line interface for interactive inference and batch processing.
- SAR-specific data augmentation: Developed specialized augmentation techniques including speckle noise simulation, elastic deformation, and realistic cutout operations that preserve label integrity while increasing effective dataset size (consisting of just near land or coastal scenarios) from 393 to 2000+ training images.
- Web application interface: Developed a separate FastAPI backend with interactive frontend for user-friendly inference, providing a web-based interface for SAR image upload, inferencing, and result visualization as a proof of concept (PoC).
- GitHub repositories: Sentinel-1 SAR Ship Detection, Wavewatch AI Portal

6 Workflow Diagram

This section outlines the key steps of our ship detection model creation process, from initial data processing to final output generation.

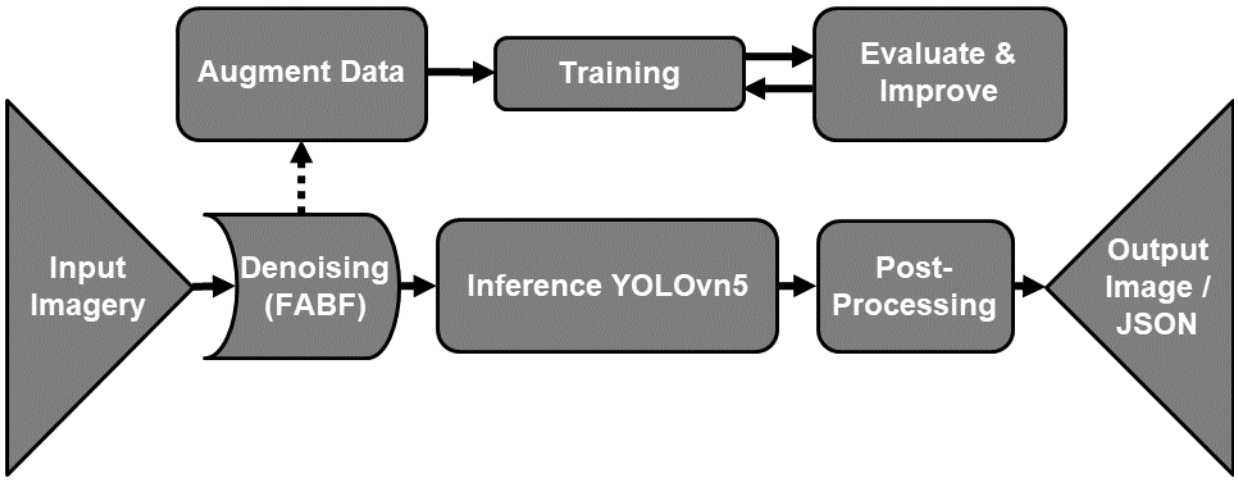
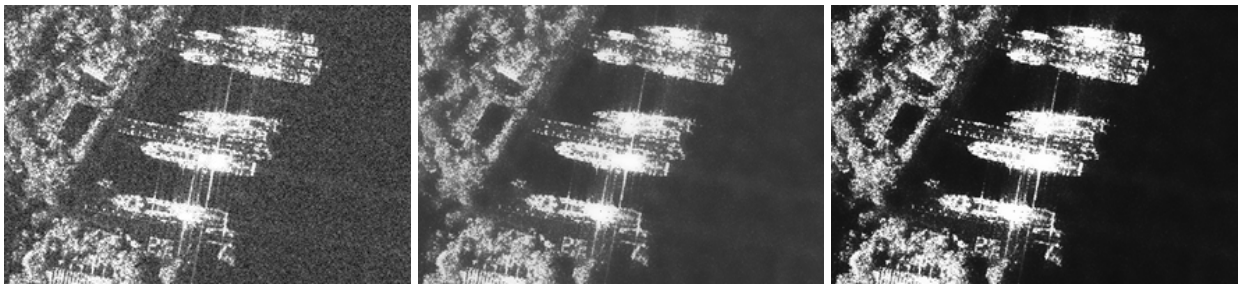


Figure 5: General workflow diagram of the project

The upper branch of the diagram illustrates the training process, where data augmentation and model analysis are performed to refine the AI. The main line shows the operational pipeline, where incoming SAR images are denoised, passed through the YOLOv5n model, and post-processed into the final output.

7 Image Preprocessing

7.1 Speckles, Grain & Noise



(a) Grainy (Initial) (b) Cleaned w/ FABF (Mid) (c) Clean, High contrast (Final)

Figure 6: Progression of the denoising process. (a) Shows the original noisy SAR image; (b) Shows the middle stage of the image - after the FABF alteration; (c) Shows the final form of the image before inferencing - raising the contrast accentuates ship edges.

7.1.1 Source and nature of speckle noise in SAR imagery

SAR images are inherently affected by distinct granular disturbances and noise (speckles). When the transmitted radar pulse hits the ground, it might meet numerous microscopic scatterers –

sands, small waves, etc. Each of these scatterers returns a tiny echo that is slightly delayed and phase-shifted depending on its position. The echoes from all scatterers in the cell superimpose when they return to the radar; due to their random relative phases, this superposition can be constructive (strengthening the signal) or destructive (weakening the signal), resulting in random bright and dark grainy patterns. This in turn reduces contrast and complicates interpretation, obscuring important features in the image. (Wei et al., 2022)

7.1.2 Removing speckle artifacts.

To address the granular speckle noise present in raw SAR imagery, we apply a specialized version of Fast Adaptive Bilateral Filtering (FABF) - an edge-preserving smoothing algorithm. Unlike basic Gaussian or median blurs, FABF adapts its filtering strength based on both the local contrast of pixels and the statistical range of intensities. This allows it to suppress grain without washing out important structural details such as ship contours or coastlines.

Basically, each pixel is examined in the context of its neighbourhood - a window of size $\rho = 3$. The minimum (α) and maximum (β) intensities are computed, and their difference ($\Delta = \beta - \alpha$) defines the dynamic range in that window. This range governs how strongly the pixel should be smoothed: in flat regions (low Δ), noise is suppressed aggressively; near edges (high Δ), smoothing is restricted. Instead of computing a full bilateral filter (which is computationally expensive), FABF approximates the weighting function with a polynomial expansion up to order $N = 1$.

Parameter choice:

$\rho = 3$: defines the spatial extent of smoothing (window size). Given Sentinel-1 SAR imagery has a resolution of 5x5 m/px, each pixel may represent a significant portion of a ship. Thus the default $\rho = 3$ was chosen to avoid excessive blurring across ship boundaries.

$\sigma = 0.115$: controls how strongly pixels of different intensities influence each other (contrast sensitivity). Larger values result in more aggressive smoothing but weaker edge preservation and vice versa. The chosen default value $\sigma = 0.115$ was determined experimentally, offering the best compromise between noise suppression and edge clarity for ship detection.

$N = 1$: Determines the order of the polynomial expansion used to approximate the bilateral filter. Higher N values yield substantially better approximation but severely increase computational cost. In our tests, $N = 1$ provided a good balance, since the improvement from higher orders was marginal for the sea specifically, which is the main focus.

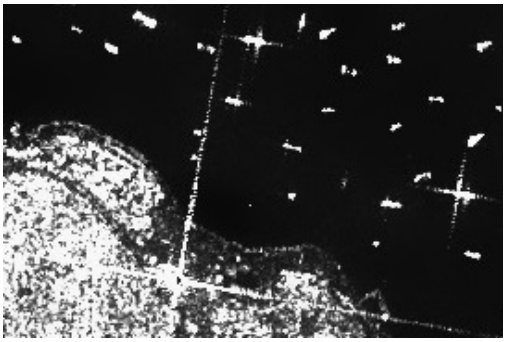
The effect is that uniform regions are smoothed heavily, eliminating granular noise, while edges and boundaries remain intact, preserving the features that are essential for downstream ship detection. (Wei et al., 2022)

Algorithm 1 Fast Adaptive Bilateral Filtering (FABF)

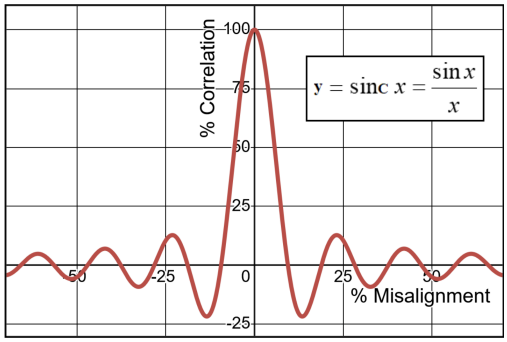
function FABF($image, \rho = 3, \sigma = 0.115, N = 1$)
  for each pixel Px in $image$ **do**
    Define a local window of radius ρ
    $\alpha \leftarrow$ minimum intensity in window
    $\beta \leftarrow$ maximum intensity in window
    $\Delta \leftarrow \beta - \alpha$ ▷ captures local contrast
    Compute polynomial expansion up to order N
    Estimate adaptive coefficients from local statistics
    if Δ is small **then**
      $Px \leftarrow \alpha + \Delta \times WeightedAverage(coefficients, \sigma)$
    end if
  end for
  return denoised image
end function

Although FABF successfully removes speckle noise and produces a smooth and clean image, suppression of local fluctuations can cause the ocean surface to appear brighter, effectively 'melting' the speckles into the background. This reduces the visual distinction between ships and the sea. To mitigate this, a subsequent contrast enhancement step is applied. The contrast operation makes bright areas brighter and dark areas darker, sharpening edges and bringing out important structures such as ship hulls and coastlines. The combination of FABF denoising followed by contrast increase therefore yields imagery that is both free of granular artifacts and well-suited for reliable detection of maritime targets. (see figure 6c)

7.2 Star-like artifacts



(a) Example of IPIs



(b) Sinc-like properties of Match function

Figure 7: (a) shows the image representation of an IPI (big one is a lighthouse, small one is a ship). (b) shows the form of the IPI artifact - if the correlation and signal strength are big enough, it appears on the image

SAR imagery is often affected by characteristic cross-shaped artifacts, also known as inter-pixel interference (IPI) patterns (see figure 7a). These arise from strong radar reflections that "spill

over" into neighboring pixels, producing bright cross-shaped sidelobes around the true target (see 7b). The effect is most pronounced near sharp edges or metallic structures like ships, which return very strong signals. In the context of ship detection, IPI artifacts are not necessarily detrimental, as they occur frequently around vessels, providing an additional cue that helps the model to locate ships in cluttered backgrounds.

There are filtering approaches, such as maximum-median filtering (MMF) and more advanced adaptive suppression techniques, that can reduce the impact of IPI artifacts. However, these methods introduce further computational overhead. In practice, much of the artifact is already mitigated by the previously applied FABF, leaving only the central part of the cross-shaped pattern. Testing even relatively lightweight methods such as MMF, any gain from further IPI suppression is negligible compared to the added complexity. It is therefore concluded that leaving IPI patterns largely untreated beyond what is already done with the FABF algorithm is a reasonable trade-off for on-edge ship detection applications without extra computational capacity.

7.3 Land Masking

It was decided to omit on-board land masking from our processing pipeline. This avoids the significant risk of accidentally masking out ships near the coast, which could create False Negatives. This approach also drastically reduces the computational load on the satellite. It is far more efficient and reliable to filter out any False Positives (e.g., a piece of land incorrectly flagged as a ship) during ground-based post-processing, where computational resources are not a constraint. This prioritizes finding every potential ship and ensures a faster, more robust system.

7.4 Incorporating Denoising in the Pipeline

The FABF (Fast Adaptive Bilateral Filtering) denoising module has been seamlessly integrated into the YOLOv5 pipeline through a modular wrapper system. The integration occurs at the data loading stage, where denoising parameters are configurable through hyperparameter files and can be selectively applied during both training and inference.

The system implements a registry-based approach where denoising methods are registered and can be dynamically selected. The FABF denoiser wraps the core `adaptive_bilateral_filter` function, implementing a standardized `DenoisingMethod` interface that ensures compatibility with the existing pipeline. During data loading, images are automatically converted from BGR to RGB, processed through the FABF algorithm with configurable parameters (rho, N, sigma, theta), and converted back to BGR for OpenCV compatibility.

Key integration points include:

- DataLoader Integration: Denoising is applied during image loading in `LoadImagesAndLabels` class
- Parameter Configuration: Denoising parameters are loaded from YAML hyperparameter files
- Format Handling: Automatic color space conversion between BGR/RGB during processing
- Conditional Application: Denoising can be enabled/disabled and applied with configurable probability

The FABF denoising module is integrated into the YOLOv5 pipeline at the data loading stage. Denoising parameters are configurable and can be enabled during training or inference. A registry pattern allows dynamic selection of different denoising methods while maintaining a consistent interface.

Algorithm 2 Denoising Integration Pseudocode

```
Define interface DenoisingMethod with method denoise(image, params)
Registry stores available denoising methods
Register FABF method with parameters {rho, N, sigma, theta, clip}

procedure ApplyDenoising(image, params)
    if params.enabled then
        denoiser ← Registry.get(params.method)
        image ← FABF.denoise(image, params)
    end if
    return image
end procedure

procedure LoadImage(index)
    image ← load_raw_image(index)
    image ← ApplyDenoising(image, denoise_params)          ▷ Integration point
    image ← letterbox(image, 640, 640)
    return tensor(image), labels
end procedure
```

8 AI Model: Methodology and Implementation

8.1 Input/Output Architecture

8.1.1 Model Input Specifications

The YOLOv5n model accepts SAR imagery processed through a standardized preprocessing pipeline optimized for object detection. The input follows these specifications:

- Input Dimensions: Images are resized to a fixed 640×640 resolution using the letterbox algorithm, which preserves aspect ratio by scaling and applying zero-padding (default padding value: $(114, 114, 114)$).
- Image Format: Sentinel-1 SAR images are processed as multi-channel inputs. The system automatically handles format conversion, ensuring three-channel compatibility with YOLOv5. Input precision is 8-bit per channel, with automatic BGR to RGB conversion during preprocessing.
- Batch Format: Batches are tensors of shape $(B, 3, H, W)$ where B is the batch size and $H = W = 640$. Images are normalized to $[0, 1]$ by dividing by 255 and converted to contiguous PyTorch tensors for efficient GPU processing.
- Data Augmentation: During training, the pipeline applies geometric transformations (rotation, scaling, translation, shear, perspective), HSV colour adjustments, and advanced augmentations such as mosaic (combining four images) and mix-up (image blending). Additionally, SAR-specific augmentations include speckle noise simulation, Gaussian blur, unsharp masking, and elastic deformation to improve robustness against SAR imaging artifacts and varying ship scales in coastal scenes.

The preprocessing pipeline is compatible with Sentinel-1 Ground Range Detected (GRD) products and supports automatic format conversion where required. It ensures consistency across training, validation, and inference while maintaining compatibility with standard YOLOv5 workflows.

8.1.2 Model Output Architecture

The YOLOv5n model produces structured detection outputs in the standard YOLO format:

- Bounding Box Coordinates: $(x_{center}, y_{center}, width, height)$ in normalized coordinates $(0, 1)$ relative to the input image dimensions.

- Confidence Score: Objectness probability $P_{obj} \in [0,1]$ indicating the likelihood of ship presence.
- Class Probability: Classification confidence $P_{cls} \in [0,1]$ (single-class detection: ship).
- Final Score: Combined confidence $P_{final} = P_{obj} \times P_{cls}$ used for thresholding.

Predictions are generated at three scales corresponding to different downsampling factors in the feature pyramid network. The YOLOv5n architecture uses three detection heads with feature maps at different resolutions: P3 (8× downsampling), P4 (16× downsampling), and P5 (32× downsampling). Each scale employs predefined anchor boxes with different aspect ratios to capture ships of varying sizes.

Post-processing applies Non-Maximum Suppression (NMS) with configurable thresholds (default: IoU = 0.45, confidence = 0.25) to remove redundant detections. The final output for each detected ship is:

$$\text{Detection} = \texttt{[class\_id, x\_center, y\_center, width, height, confidence]} \quad (2)$$

where $class\_id = 0$ for the ship class in this project. The system supports multiple output formats including normalized coordinates for training labels and pixel coordinates for visualization. The streamlined output enables direct integration into downstream packaging and visualization pipelines, while maintaining real-time performance on resource-constrained hardware.

8.2 Training Pipeline

This section describes the training pipeline used to develop the ship detection model from Sentinel-1 SAR imagery using the YOLOv5n architecture. The pipeline was designed to achieve robust detection accuracy while maintaining real-time inference performance on the NVIDIA Jetson AGX Orin platform. The training strategy emphasizes generalization across sea states, sensor variations, and ship scales, while keeping the model lightweight for embedded deployment.

8.2.1 Data Augmentation

The SAR augmentation system implements domain-specific transformations designed for SAR imagery while maintaining label integrity. The pipeline combines intensity-based augmentations with geometric transformations to increase dataset diversity. The pipeline can be seen in Algorithm 3. Table 3 mentions the hyperparameters used for data augmentation.

Parameter	Value	Description
cutout_prob	0.3	Cutout augmentation probability
noise_prob	0.2	Speckle noise probability
blur_prob	0.15	Gaussian blur probability
elastic_prob	0.1	Elastic deformation probability

Table 3: Augmentation hyperparameters

Augmentation Types Intensity Augmentations (Label-Preserving):

- Speckle Noise: Multiplicative exponential noise simulation with additive Gaussian noise
- Gaussian Blur: Kernel sizes 3×3, 5×5, or 7×7 with $[0.5, 1.5]$
- Unsharp Masking: Edge enhancement using 1.5×original - 0.5×blurred

Geometric Augmentations (Label-Transforming):

- Cutout: Random rectangular regions (5-15% of image) filled with shadow intensity [20, 60]
- Elastic Deformation: Displacement fields with $[5, 15]$ pixels, smoothed with 15×15 Gaussian kernel
- Synthetic Samples: Horizontal/vertical flips and small rotations ($\pm 10°$)

Label Handling All augmentations maintain YOLO format compatibility [class, x_center, y_center, width, height]. Geometric transformations use OpenCV's perspective transform matrices applied to bounding box corner points, ensuring accurate coordinate updates.

Algorithm 3 SAR Augmentation Pipeline

Input: image I, labels L, hyperparameters H
$I_{aug} \leftarrow I.copy()$, $L_{aug} \leftarrow L.copy()$
if $random() < H.noise\_prob$ **then**
    $I_{aug} \leftarrow add\_speckle\_noise(I_{aug})$
end if
if $random() < H.cutout\_prob$ **then**
    $I_{aug}, L_{aug} \leftarrow apply\_cutout(I_{aug}, L_{aug})$
end if
if $random() < H.elastic\_prob$ **then**
    $I_{aug}, L_{aug} \leftarrow apply\_elastic\_deformation(I_{aug}, L_{aug})$
end if
Return: I_{aug}, L_{aug}

Algorithm 4 Synthetic Sample Creation

for each sample i in $n\_samples$ **do**
  if $i = 0$ **then**
    $I_{syn} \leftarrow flip\_horizontal(I)$
    $L_{syn} \leftarrow flip\_labels\_horizontal(L)$
  else if $i = 1$ **then**
    $I_{syn} \leftarrow flip\_vertical(I)$
    $L_{syn} \leftarrow flip\_labels\_vertical(L)$
  else
    $\theta \leftarrow random(-10, +10)$
    $I_{syn}, L_{syn} \leftarrow rotate\_simple(I, L, \theta)$
  end if
  if $|L_{syn}| > 0$ **then**
    $samples.append((I_{syn}, L_{syn}))$
  end if
end for

Label Validation Transformed labels undergo boundary validation to ensure they remain within image bounds with 5% margin. Labels heavily obscured by cutouts (>60% intersection) are filtered out to maintain training quality.

Figure 8 gives an overview of the data augmentation, which is being performed in the pipeline.

8.2.2 Loss Functions

The loss function combines three terms:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{box}} \cdot \mathcal{L}_{\text{box}} + \lambda_{\text{obj}} \cdot \mathcal{L}_{\text{obj}} + \lambda_{\text{cls}} \cdot \mathcal{L}_{\text{cls}}$$

$\mathcal{L}_{\text{box}}$: Efficient IoU (EIoU) loss, which penalizes bounding box misalignment by considering overlap, center distance, and aspect ratio. $\mathcal{L}_{\text{obj}}$: Binary cross-entropy (BCE) loss for objectness confidence. $\mathcal{L}_{\text{cls}}$: BCE loss for classification confidence (single class).

The weighting factors are $\lambda_{\text{box}} = 0.05$, $\lambda_{\text{obj}} = 1.0$, and $\lambda_{\text{cls}} = 0.5$, reflecting the priority of objectness over classification in single-class ship detection.

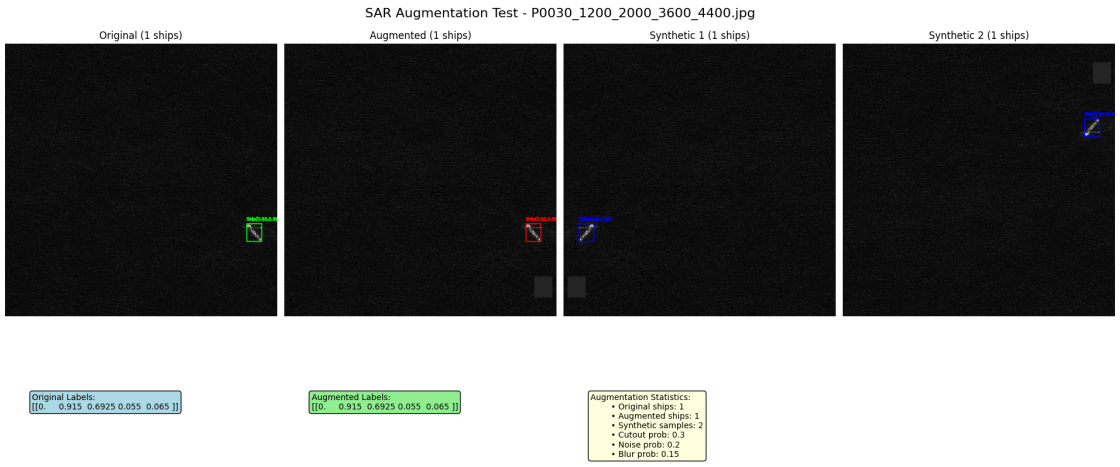


Figure 8: Comprehensive SAR augmentation pipeline validation showing: (a) Original SAR image with ship detection (green bounding box), (b) Augmented image with SAR-specific transformations including cutout augmentation (red bounding box), and (c-d) Synthetic samples demonstrating geometric transformations through horizontal/vertical flips and small rotations (blue bounding boxes). The pipeline successfully maintains label integrity while applying domain-specific augmentations.

8.2.3 Optimizer and Scheduler

Training employed stochastic gradient descent (SGD) with Nesterov momentum and cosine annealing learning rate scheduling with warmup. This configuration balanced convergence stability and computational efficiency for embedded deployment.

Parameter	Value
Optimizer	SGD with Nesterov momentum
Initial Learning Rate	0.01
Momentum	0.937
Weight Decay	0.0005
Scheduler	Cosine annealing with 3-epoch warmup

Table 4: Optimizer and scheduler configuration.

Learning Rate Scheduler: The model employs a cosine annealing learning rate schedule with warmup, implemented through a custom one-cycle function. The scheduler gradually reduces the learning rate from its initial value toward a final multiplier over the training duration:

$$\mathrm{LR}(t) = \mathrm{lr}_0 \cdot \left(\frac{1 + \cos(\pi \cdot t/T)}{2} \right) \cdot (1 - \mathrm{lrf}) + \mathrm{lrf} \tag{3}$$

where t is the current step, T is the total number of steps, $\mathrm{lr}_0 = 0.01$ is the initial learning rate, and $\mathrm{lrf} = 0.01$ is the final learning rate multiplier. The warmup phase spans 3 epochs with initial momentum of 0.8 and bias learning rate of 0.1, providing stable initialization before the main training phase.

8.2.4 Hyperparameters

Hyperparameters were defined via structured YAML files and tuned for SAR ship detection. Key values are summarized in Table 5.

Parameter	Value	Description
lr0	0.01	Initial learning rate
lrf	0.01	Final LR multiplier (cosine decay)
momentum	0.937	SGD momentum term
weight_decay	0.0005	L2 regularization factor
warmup_epochs	3.0	Warmup duration
warmup_momentum	0.8	Momentum during warmup
warmup_bias_lr	0.1	Initial bias LR
box	0.05	Loss gain multiplier for box regression
cls	0.5	Loss gain multiplier for classification
obj	1.0	Loss gain multiplier for objectness
iou_t	0.20	IoU threshold for positive anchor assignment
anchor_t	4.0	Anchor-matching threshold

Table 5: Selected training hyperparameters.

Inference Filtering Pipeline After the forward pass through the model, each image produces raw bounding box predictions in the format `[class_id, x_center, y_center, width, height, confidence]`. These predictions undergo the following sequential filtering stages:

1. Confidence Thresholding: Predictions with objectness scores below the configurable threshold (`conf_thres = 0.25`) are discarded. This constitutes the primary filtering mechanism for reducing false positives in cluttered maritime SAR imagery.

2. Class Filtering (Optional): Detections can be filtered by class index if enabled via commandline interface or API configuration. For single-class ship detection, this defaults to class 0, while maintaining extensibility for multi-class scenarios.

3. Bounding Box Conversion: Coordinates are transformed from YOLO format (c_x, c_y, w, h) to

pixel format (x_1, y_1, x_2, y_2) using the `xywh2xyxy()` function, preparing detections for standard NMS operations and spatial analysis.

4. Confidence Scoring: The final detection confidence is computed as:

$$\text{final\_conf} = \text{objectness\_score} \cdot \text{class\_probability} \tag{4}$$

   This multiplicative approach balances spatial localization certainty with semantic classification confidence.

5. Non-Maximum Suppression (NMS): Bounding boxes are ranked by confidence scores and overlapping detections with IoU exceeding the threshold (`iou_thres = 0.45`) are suppressed using PyTorch's `torchvision.ops.nms()` implementation. This retains only the highest-scoring detection per spatial region, with a maximum of 300 detections per image (`max_det = 300`).

Post-NMS processing includes rescaling detections from the inference resolution (640×640) back to the original SAR image dimensions using the `scale_boxes()` function, ensuring accurate spatial localization in the original image coordinate system.

8.2.5 Inference Filtering

After inference, predictions undergo the following filtering steps:

1. Confidence Thresholding: Remove detections below a confidence threshold (default 0.25).
2. Bounding Box Conversion: Transform predictions to pixel coordinates.
3. Non-Maximum Suppression (NMS): Suppress overlapping boxes above IoU threshold (default 0.45).

The final outputs are rescaled to the original SAR image dimensions. If no detections remain, the image is classified as ship-free and excluded from results.

8.2.6 Configurable Parameters

The detection pipeline allows configuration of thresholds and maximum detections:

Parameter	Default	Description
`conf_thres`	0.25	Confidence threshold
`iou_thres`	0.45	IoU threshold for NMS
`max_det`	300	Maximum detections per image

Table 6: Configurable inference parameters.

9 Software Architecture

The code architecture implements a modular, configuration-driven design pattern optimized for SAR ship detection workflows. The implementation utilizes YAML-based model definition files in `models/` for flexible architecture composition, with custom attention mechanisms (`models/rfa.py`) and comprehensive loss functions (`utils/loss.py`) implemented as pluggable modules.

The training pipeline (`train.py`) orchestrates data loading, model instantiation, and optimization, while the inference engine (`detect.py`) provides streamlined detection capabilities with configurable post-processing parameters. The system includes a comprehensive pipeline framework (`utils/pipeline/`) detailed in figure 10 for end-to-end SAR data processing, including ingestion, georeferencing, and result packaging.

Demo Pipeline The SAR ship detection pipeline processes Sentinel-1 synthetic aperture radar imagery through five sequential stages: ingest validation, YOLO-based ship detection, georeferencing, post-processing with thumbnail generation, and final packaging for downlink. The system employs (optionally - i.e. the user can request to denoise the images or not) fast adaptive bilateral filtering for SAR denoising, YOLOv5 for vessel detection, and (mock) automated coordinate transformation to convert pixel coordinates to geographic coordinates. Failed processing stages are logged and quarantined, while successful detections progress through the workflow to generate compact downlink packets suitable for satellite transmission constraints.

The pipeline generates comprehensive ship detection outputs packaged into compressed downlink packets. Each packet includes:

1. Detection Results – YOLO bounding box coordinates with confidence scores and class IDs.
2. (Mock) Georeferenced Data – geographic coordinates (latitude/longitude), WKT footprints, and pixel-to-geo transformations. Intended for possible future extensions involving real georeferenced images.
3. Thumbnail Images – 256×256 pixel cropped regions around detected vessels for rapid visual verification.
4. Metadata – scene identifiers, acquisition timestamps, image dimensions, and geospatial bounds.

5. Processing Logs – confidence thresholds, detection counts, and quality metrics.

6. JSON Detection Files – structured data containing all ship detection details, coordinates, confidence scores, vessel classifications, and geospatial information in machine-readable format for automated processing and analysis.

The packager compresses these outputs into ≤ 10 MB ZIP archives with SHA-256 checksums, prioritizing recent detections and retrying failed transmissions. Each packet also contains a `manifest.json` cataloging all contents for downstream ground-station validation.

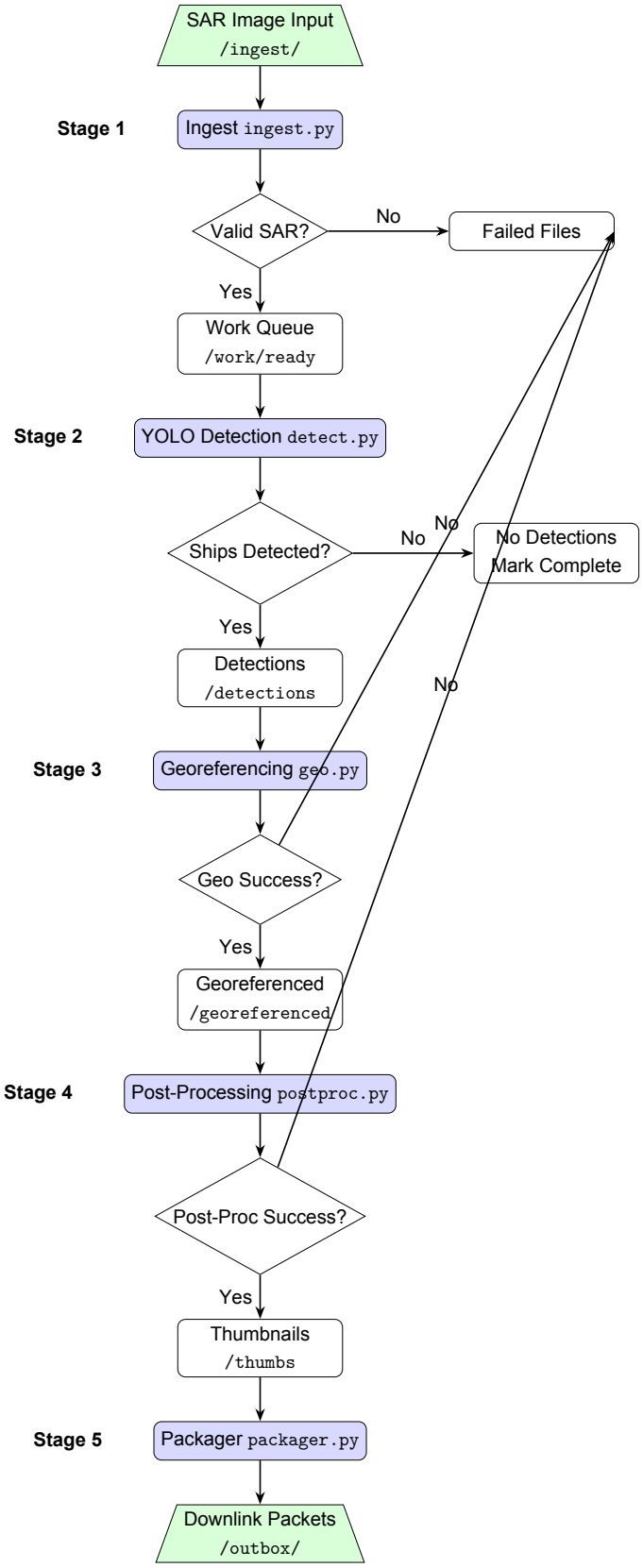


Figure 10: Compact SAR ship detection pipeline flow (vertical layout).

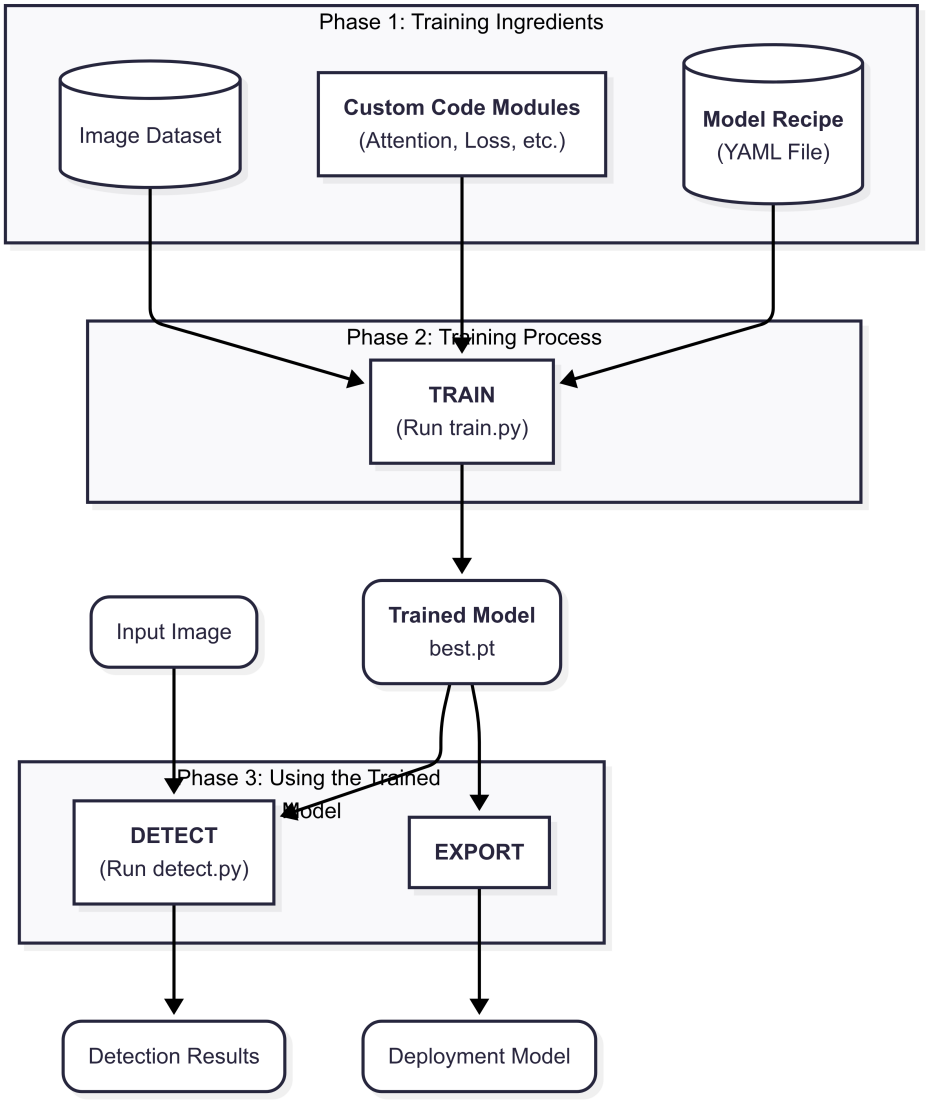


Figure 9: Three-phase machine learning pipeline for SAR ship detection. Phase 1 (Training Ingredients) combines image datasets, custom attention modules (RFCBAMConv, C3_CA), and YAML model configurations. Phase 2 (Training Process) executes the training workflow via `train.py` to produce optimized model weights. Phase 3 (Deployment) bifurcates into detection inference (`detect.py`) for real-time ship identification and model export for embedded deployment on Jetson devices.

10 Model Evaluation, Results, and Improvement

This section details the complete process of evaluating our YOLOv5n ship detection models to select the most effective configuration. A comparative study of the four models that were trained is shown, explaining the rationale behind our final selection using key performance metrics. Following these results, an in-depth analysis of the chosen model is conducted to determine its optimal operational parameters and discuss other key architectural decisions made during development.

10.1 Experimental Setup

We train the YOLOv5n model from scratch on different variants of the HRSID dataset as discussed in the following section for 350 epochs.

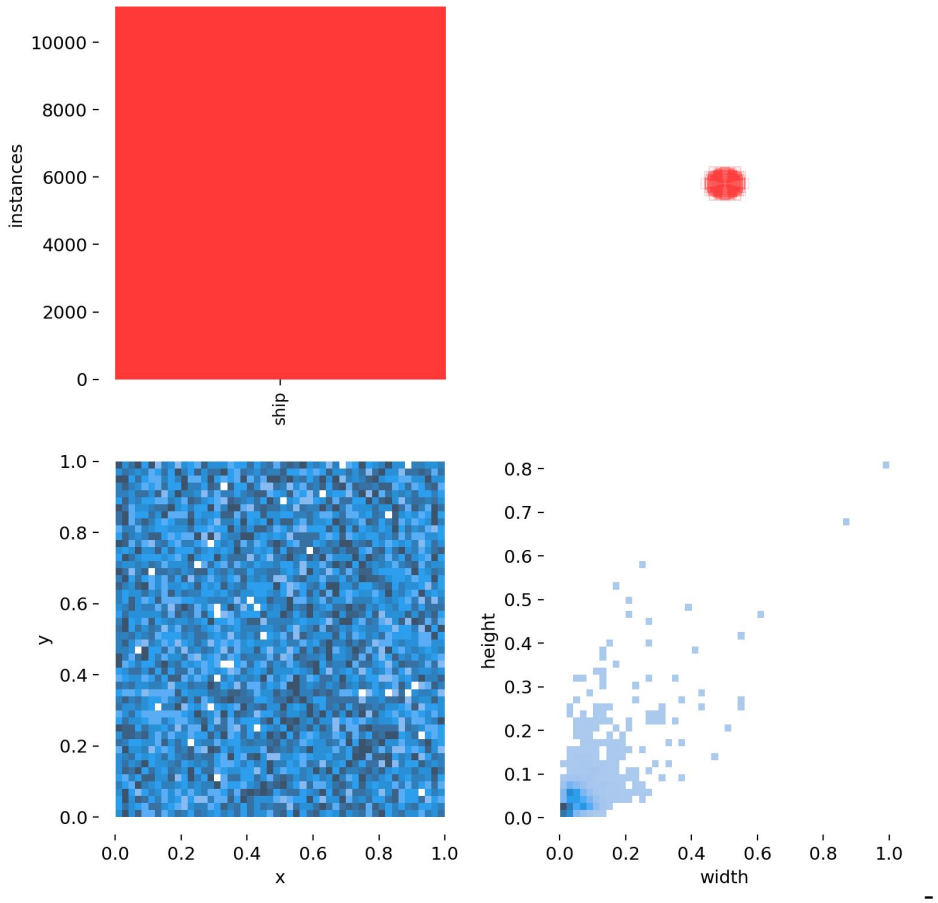
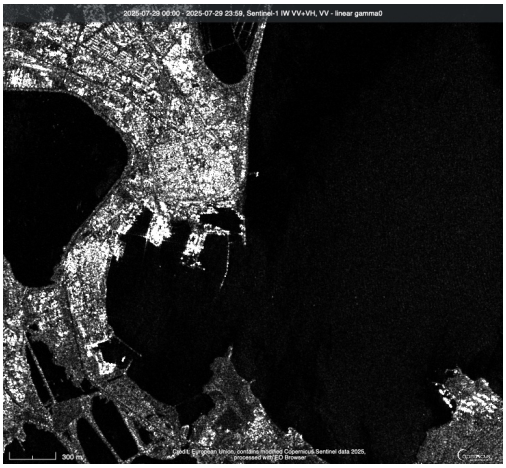


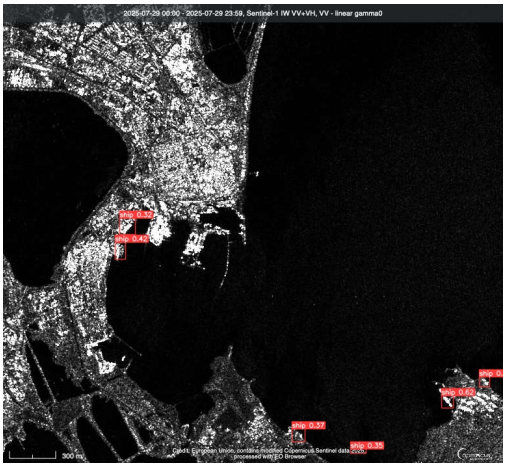
Figure 11: Label distribution visualization from the training dataset. Top-left: instance count per class. Top-right: heatmap of bounding box center locations. Bottom-left: spatial distribution of (x, y) centers. Bottom-right: distribution of bounding box dimensions (width, height).

Figure 11 visualizes the spatial and dimensional characteristics of the annotated ship targets in

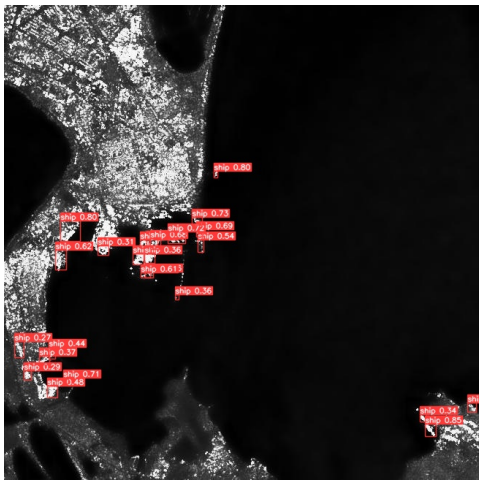
the training dataset. The top-left bar plot confirms that the dataset contains a single object class (`ship`) with over 10,000 instances. The top-right heatmap illustrates the density of bounding box center points, revealing a tight spatial clustering suggestive of scene-specific concentration or limited imaging swaths. The bottom-left 2D histogram plots the (x, y) coordinates of bounding box centers, showing a relatively uniform spatial spread across the image domain. The bottom-right panel visualizes the joint distribution of bounding box width and height, which is strongly skewed toward small object sizes - consistent with typical SAR ship detection scenarios involving distant, high-resolution observations.



(a) Input SAR image (no detections)



(b) YOLOv5n output: noisy image



(c) YOLOv5n output: denoised image

Figure 12: Qualitative comparison of detection outputs on the same SAR image, upon some denoising using fast adaptive bilateral filtering.

Figure 12 presents a qualitative result of YOLOv5n inference on a dense maritime scene. YOLOv5n. When tested against bigger models, there appears to be a trade-off between computational efficiency and architectural complexity. Given the memory constraints, and the requirement of deployment

on embedded devices, YOLOv5n seems to be a good choice, even with its smaller size.

10.2 Model Comparison and Selection

This project trained four different models to find the best strategy, focusing on two key variables: the composition of the training data (a mix of sea/land vs. mainly land) and the use of a denoising preprocessing step.

10.2.1 Test Strategy and Metrics

In order to compare the models, they were tested on separate Land and Sea datasets. This is crucial because a single score from a "mixed" dataset could be misleading - strong performance at sea could hide major weaknesses near complex coastlines. The separate testing gives a much clearer picture of each model's real-world capabilities.

In order to find the best model a single metric was needed, which could capture the balance between two critical goals: finding all the real ships (Recall) and not raising too many false alarms (Precision). Based on these criteria the F1-Score metric was chosen, a harmonic mean of both, where a higher score indicates a better overall balance. The table below shows the peak F1-Score for each model and the optimal confidence threshold (the model's minimum certainty level for a detection) required to achieve it.

Model	Land Noise Dataset	Sea Dataset	Sea Denoised Dataset	Land Denoised Dataset
Full Denoised	**0.803 (0.20)**	**0.979 (0.20)**	**0.979 (0.20)**	**0.825 (0.45)**
Land Denoised	0.779 (0.45)	0.976 (0.20)	0.976 (0.20)	0.782 (0.35)
Land Noise	0.795 (0.50)	0.983 (0.30)	0.983 (0.20)	0.731 (0.55)
Full Noise	0.850 (0.40)	0.979 (0.30)	0.979 (0.20)	0.756 (0.55)

Table 7: F1-Score performance of all models across the four test sets. The chosen model configuration is highlighted in bold.

10.2.2 Final Choice: The ‘Full Denoised‘ Model

Although the ‘Full Noise‘ model achieved the highest F1-Score on specific ‘Land Noise Dataset‘, the final choice is the Full Denoised model. This decision is based on a broader consideration of real-world deployment, generalization, and model stability.

Performance on a single benchmark may not fully predict a model's effectiveness in a live environment with diverse and unpredictable conditions. Real-world SAR data contains many different types of noise from various sensors and sea states. The Full Denoised approach creates a more

stable and predictable pipeline by standardizing the input. By learning from a cleaner signal, the model is expected to be more robust and generalize better to unseen types of noise.

This strategy involves a clear trade-off: a slightly lower F1-score was accepted on this specific test and a longer processing time due to the denoising step. However, it is believed that this is a worthwhile investment for the expected increase in reliability in a real-world system where stability is paramount.

10.3 In-Depth Analysis of the Chosen Model

This section takes a closer look at the 'Full Denoised' model to select its final, unified confidence threshold. To understand its behaviour in detail, there will be an examination in its performance on both the Land Denoised and Sea Denoised datasets, using Precision-Recall (P-R) curves for a visual overview and tables for the precise metrics.

10.3.1 Performance in a Coastal Environment

The Land Denoised Dataset represents the most complex environment for the chosen model. A Precision-Recall curve is the best way to visualize the trade-off the model must make between finding all ships (Recall) and ensuring its detections are correct (Precision).

The curve in Figure 13 plots this relationship. A threshold that provides a strong balance is desired. This is typically found at the kink of the curve before recall drops sharply for minimal gains in precision.

The visual analysis suggests that a threshold around 0.45 is a robust choice. Table 8 provides the exact numerical data for this trade-off at various thresholds, confirming the visual interpretation.

Threshold	TP	FP	FN	Precision	Recall	F1-Score
...	...	...	...	...	...	...
0.40	351	69	82	83.57%	81.06%	0.823
0.45	**340**	**51**	**93**	**86.96%**	**78.52%**	**0.825**
0.50	327	41	106	88.86%	75.52%	0.816
...	...	...	...	...	...	...

Table 8: Performance of the 'Full Denoised' Model on the Land Denoised Dataset.

As the curve predicted, table 8 confirms that the model achieves its peak F1-score of 0.825 at a confidence threshold of 0.45. At this point, it correctly identifies over 78% of all ships while maintaining a precision of nearly 87%.

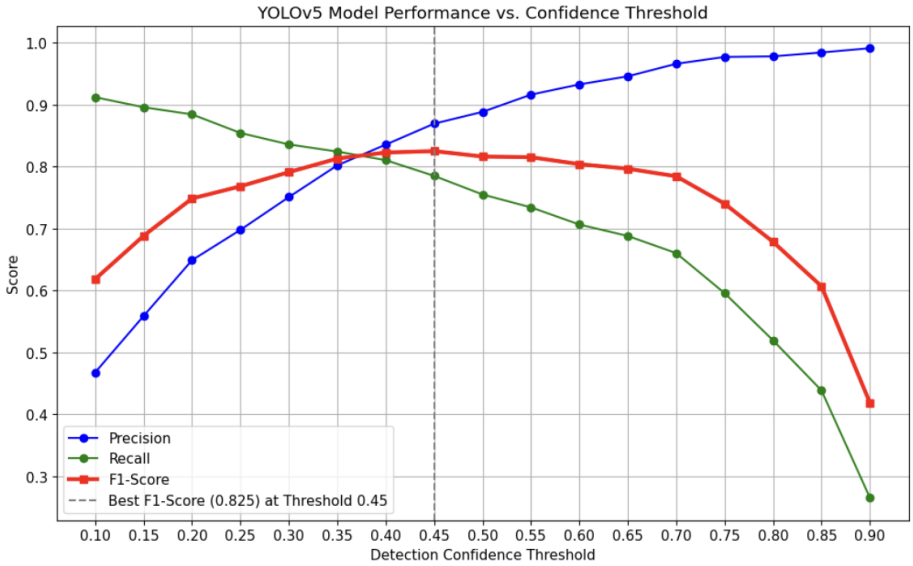


Figure 13: Precision-Recall curve for the 'Full Denoised' model on the Land Denoised dataset. The selected operational threshold of 0.45 is marked, indicating a strong balance point on the curve.

10.3.2 Performance in an Open Ocean Environment

In the less cluttered open-sea environment, the model's performance is, as expected, extremely high. The P-R curve in Figure 14 is pushed far into the top-right corner, which indicates near-perfect performance where high precision is maintained even at very high recall.

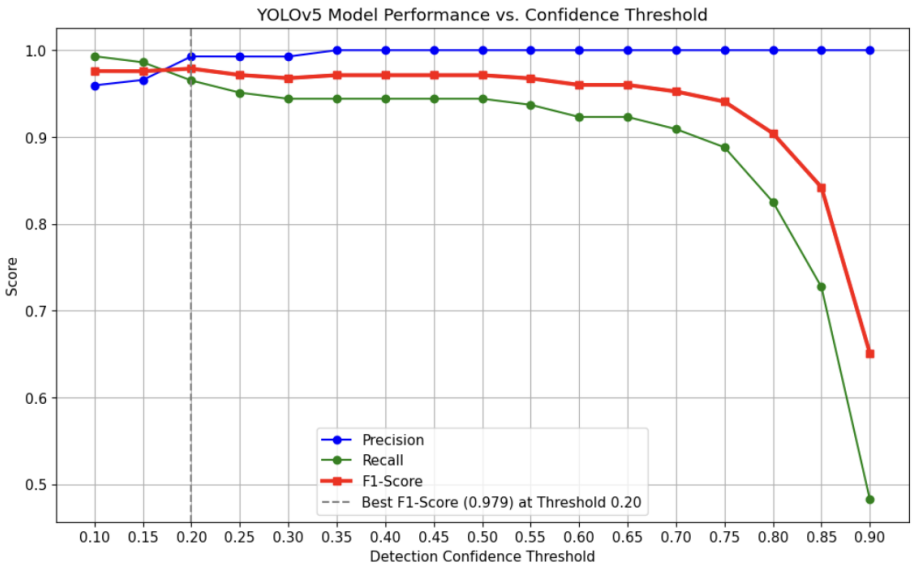


Figure 14: Precision-Recall curve for the 'Full Denoised' model on the Denoised Sea dataset. The curve's proximity to the top-right corner (the point of 100% precision and 100% recall) signifies outstanding performance.

The graph clearly shows the model's accuracy and reliability in ideal conditions. Table 9 quantifies this exceptional performance, showing that the model achieves a peak F1-score of 0.979.

Threshold	TP	FP	FN	Precision	Recall	F1-Score
...	...	...	...	...	...	...
0.20	**138**	**1**	**5**	**99.28%**	**96.50%**	**0.979**
...	...	...	...	...	...	...
0.45	**135**	**0**	**8**	**100.00%**	**94.41%**	**0.971**
...	...	...	...	...	...	...

Table 9: Performance of the 'Full Denoised' Model on the Denoised Sea Dataset.

10.3.3 Selecting a Unified Threshold

To maintain a simple operational workflow, the use of a single confidence threshold is important. Since the coastal environment is more critical, its optimal threshold was set to 0.45. When the 0.45 threshold was applied to the Sea Dataset, it yields an F1-Score of 0.971, a negligible decrease from its absolute peak. This confirms that 0.45 is a robust unified threshold that provides excellent performance in both environments.

Adaptive Thresholding Investigation

The fact that the optimal thresholds for land (0.45) and sea (0.20) was different lead to a further investigation of an adaptive system. The project tested a two-stage pipeline with a binary classifier to first label an image as "land" or "sea" and then apply the corresponding optimal threshold. However, the final F1-Score improvement was minimal and did not justify the added complexity and slower processing time. Therefore, it was decided to stick with the simpler, single-threshold model but keep the adaptive strategy as a potential future enhancement.

11 Deployment Pipeline

This section details the deployment pipeline of the SAR Ship Detection system, targeting embedded deployment on NVIDIA Jetson AGX Orin. The deployment emphasizes reproducibility, lightweight execution, and optimization for edge hardware.

The final deployment target is the Jetson AGX Orin, chosen for its superior GPU acceleration (Ampere architecture with Tensor Cores) and support for optimized inference runtimes (TensorRT). Specifications are summarized in Table **??**.

Component	Specification
CPU	12-core ARM Cortex-A78AE v8.2 64-bit
GPU	2048-core NVIDIA Ampere with 64 Tensor Cores
Memory	32 GB LPDDR5
Storage	NVMe SSD / external expansion
Inference Precision	FP32 / FP16 / INT8
Power Modes	Configurable 15–60W

Table 10: Jetson AGX Orin Specifications

11.1 Dockerized Environment

The complete pipeline is containerized using Docker. Two Dockerfiles are maintained:

- Laptop/Development: x86_64 CUDA-enabled container for model training, testing, and debugging.
- Jetson Deployment: ARM64 container based on NVIDIA JetPack runtime with PyTorch and TensorRT bindings.

The container encapsulates preprocessing, YOLOv5n inference, post-processing, and packaging modules. All data and weights are mounted as volumes to simplify updates without rebuilding the container.

11.2 Current Inference Pipeline (PyTorch)

In the current prototype, inference runs in PyTorch directly. Input SAR tiles (640×640) are passed through YOLOv5n and post-processed with Non-Maximum Suppression (NMS). Below is an example of how running the demo might look like on NVIDIA Jetson AGX Orin:

```
# Build Jetson-compatible Docker image
sudo docker build -f utils/docker/Dockerfile-arm64 -t sar-ship-detection:jetson .

# Run container with GPU access
sudo docker run -it --rm --gpus all --runtime nvidia --ipc=host \
  -v $(pwd):/workspace \
  --entrypoint python \
  sar-ship-detection:jetson demo.py
```

Inference Parameters

- Input resolution: 640 × 640
- Batch size: 1
- Precision: FP32 (default), FP16 (optional with –half flag)
- Max detections: 1000 (default), configurable via –max-det
- Confidence threshold: 0.25 (default), 0.5 (as used in the inference test inside docker container)
- IoU threshold: 0.45 (default)

For the final deliverable, the PyTorch-trained YOLOv5n weights will be optimized and executed via TensorRT, NVIDIA's high-performance inference runtime. This step is critical for meeting real-time constraints and efficient onboard execution.

Conversion Pathway

1. Export PyTorch model to `ONNX` using Ultralytics export script:

```
python export.py --weights yolov5n.pt --include onnx
```

2. Optimize ONNX model to TensorRT engine with FP16 precision:

```
trtexec --onnx=yolov5n.onnx \
        --saveEngine=yolov5n_fp16.engine \
        --fp16 --workspace=2048
```

3. Deploy TensorRT engine in the Jetson container and run inference with:

```
python detect.py --weights yolov5n_fp16.engine --source ...
```

For efficient deployment, the PyTorch-trained weights are exported as a TensorRT engine using the `export.py` script included in the repository. The export is performed with FP16 precision and limited TensorRT workspace to accommodate Jetson's memory constraints.

Performance

On x86_64 architecture with GPU acceleration, the inference performance demonstrates:

- YOLOv5n: 2.8 ms inference time, 32 ships detected
- Preprocessing: 0.3–0.4 ms per image

- NMS processing: 60–72 ms per image

The YOLOv5n model achieves approximately 357 FPS for 640×640 resolution images, with sufficient detection accuracy for SAR ship identification and efficient onboard deployment.

Image Build and Runtime

The Docker image incorporates PyTorch with CUDA support, OpenCV, and all necessary dependencies for SAR image processing. The build process ensures compatibility across different deployment environments while maintaining the complete inference pipeline. Container execution provides isolated environment management with volume mounting for model weights and dataset access. The GPU-accelerated inference pipeline processes single images or batch inputs with configurable confidence thresholds and NMS parameters for optimal detection performance.

11.3 Technology Stack

Main Python Libraries

- SAR Handling: `rasterio`, `gdal`, `pyroSAR`
- Deep Learning: `torch`, `tensorrt`, `pycuda`
- Image Processing: `opencv-python`, `numpy`, `Pillow`
- Postprocessing: `shapely`, `geopandas`, `matplotlib`
- Automation/Utils: `schedule`, `argparse`, `yaml`, `logging`

Hardware Specifications for Development

- Development: `x86_64` with 32 GB RAM, 8 GB VRAM, CUDA GPU
- Edge Deployment: NVIDIA Jetson AGX Orin
- Storage: Minimum 64 GB SD card or NVMe SSD

11.4 Scalability Concerns and Solutions

Jetson devices impose several scalability constraints, which is addressed as follows:

- Memory Limits: Use FP16/INT8 inference with batch size 1 and reduced detection caps

- Processing Throughput: Implement sliding window tiling and inference pipelining
- Power and Thermal Constraints: Activate active cooling and frequency scaling
- Data Volume: Use on-device compression and cloud syncing via `rclone`

12 Web Application Architecture and Implementation

To showcase the capabilities of our ship detection model and provide an interactive testing environment, we developed a full-stack web application you can access (WaveTrack, 2025). It is important to clarify that this application is a ground-segment demonstration platform. It is designed for user interaction and analysis on the ground and does not run on the satellite itself.

The primary purpose of this tool is to simulate how an end-user would interact with SAR imagery and the corresponding uncovering data relayed from an orbital asset. It provides an intuitive, hands-on way to test the model's performance, visualize its outputs, and understand its capabilities without requiring a complex software setup.

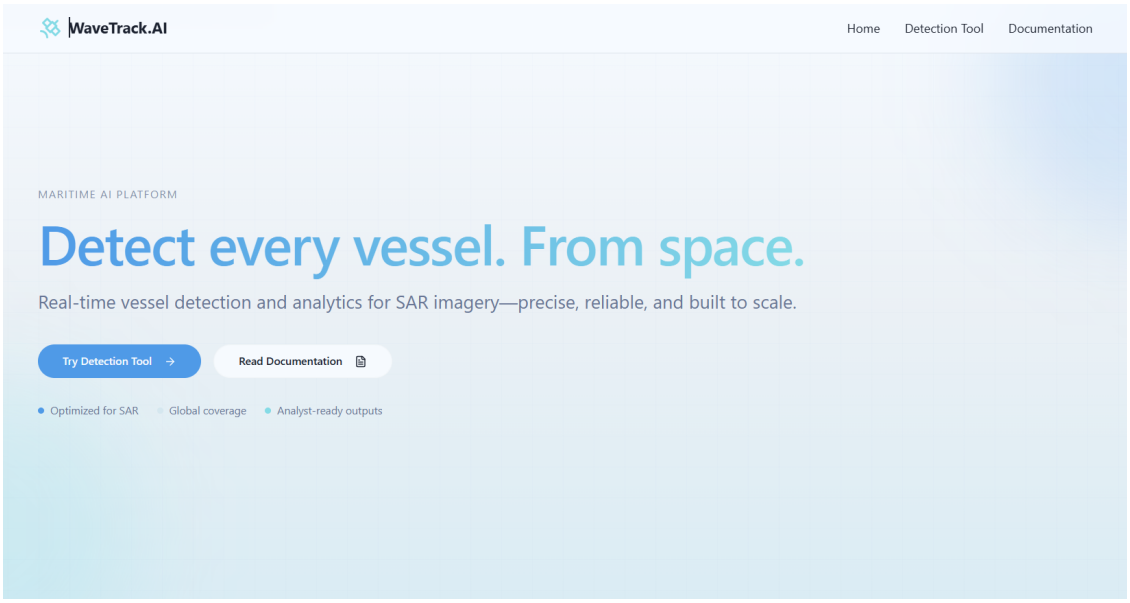


Figure 15: The main landing page for the WaveTrack platform, our ground-based detection tool.

12.1 User Experience and Workflow

The user workflow is designed to be simple and intuitive. The entire process is handled on a single, clean interface, emphasizing ease of use and rapid results.

1. Upload: The user begins by uploading a SAR image through a drag-and-drop interface. This simulates receiving a new image file from a satellite data archive.

2. Customize: Before running the analysis, the user can select their desired outputs, such as obtaining the total ship count in the image, confidence probabilities for each detection, and the processed image with bounding box overlays.

3. Analyse: The application sends the image to our backend for processing. The "Analysis Complete" status signifies that the inference has been successfully run on our ground-based server. The results are then returned and displayed in a clear, multi-panel layout, showing the original image alongside the annotated version and a list of detections.

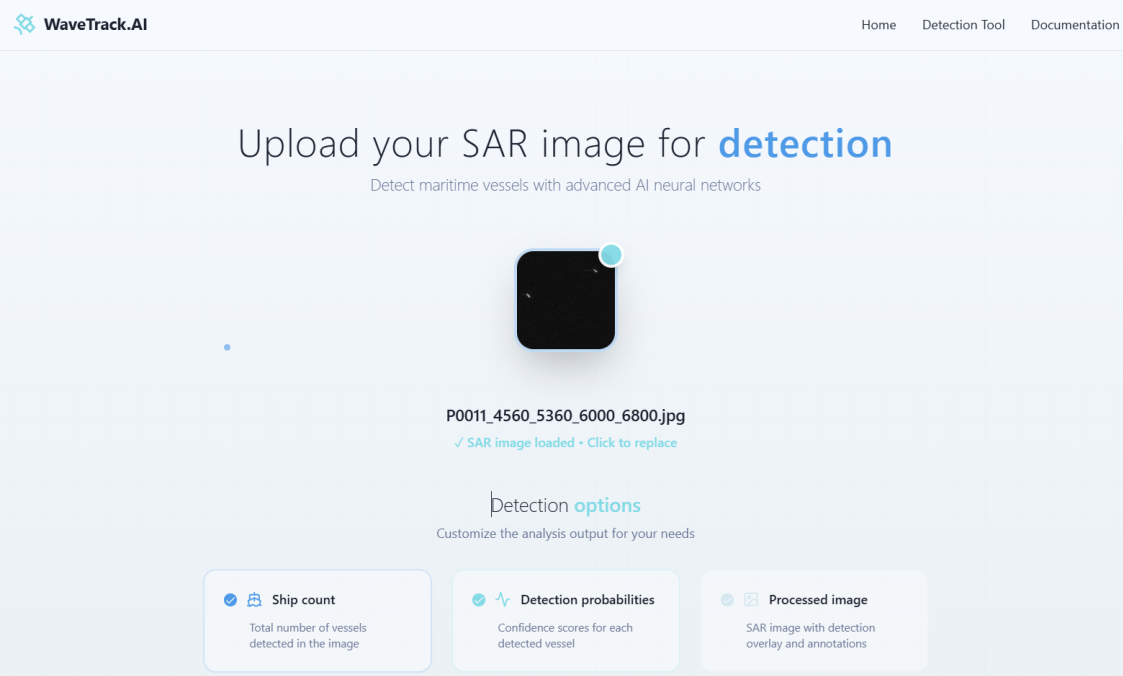


Figure 16: Image upload and desired outputs selection

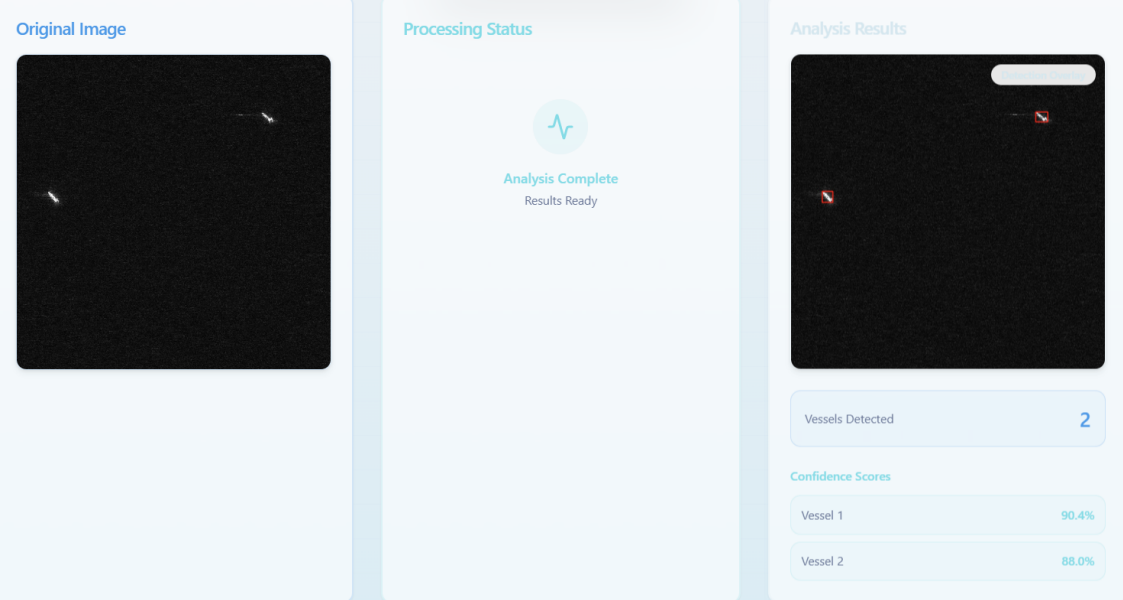


Figure 17: The end-to-end user workflow within the ground-based application, from image upload to the final analysis results.

12.2 System Architecture and Technology

The application is built with a modern technology stack, chosen to make the tool feel responsive for the user.

The system is comprised of two main components:

- Frontend: The user interface is a sophisticated single-page application (SPA) built with React and TypeScript. It runs entirely client-side in a standard web browser on the user's local machine, ensuring a fast and reliable experience. The design was rapidly developed using Tailwind CSS.
- Backend: The backend is a lightweight and efficient Python service powered by FastAPI. This API server runs on the ground and, for this demonstration, directly processes user-uploaded images to simulate the end-to-end detection pipeline. In a full operational scenario, this server would instead process data packets relayed from the satellite or fetch full images from a data archive. It uses our custom-trained YOLOv5 model on PyTorch to perform the detection.

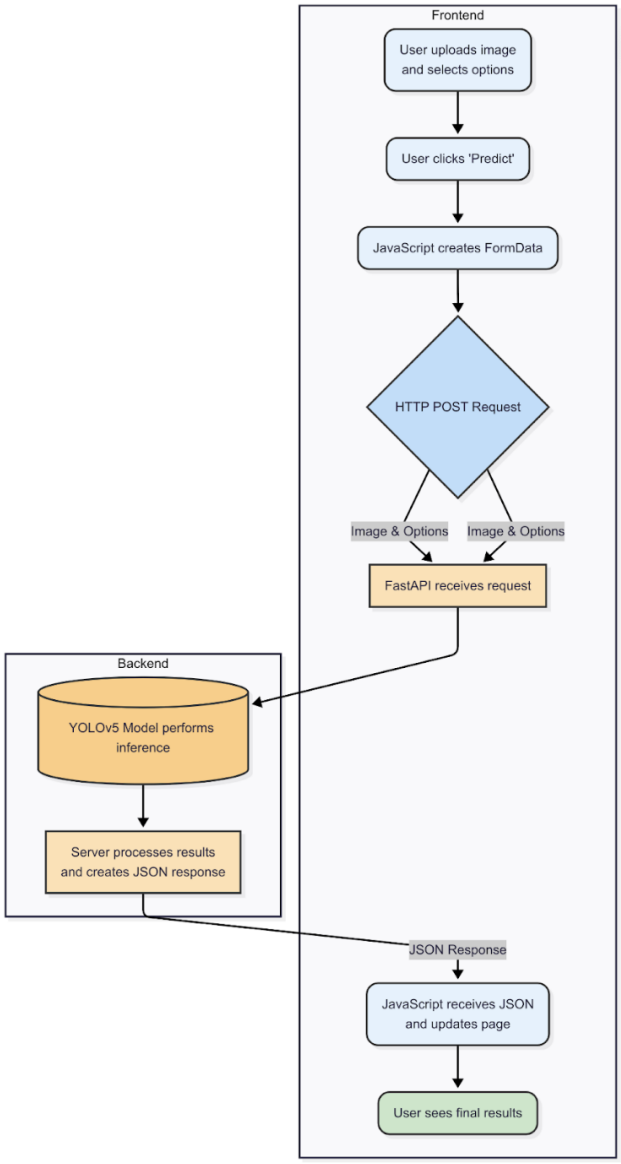


Figure 18: Web Application data workflow from frontend to backend.

12.3 Documentation and Resources

There are also links that provide access to this PDF file and the open-source GitHub repository, where the codebase for this ground application can be explored and contributed to.

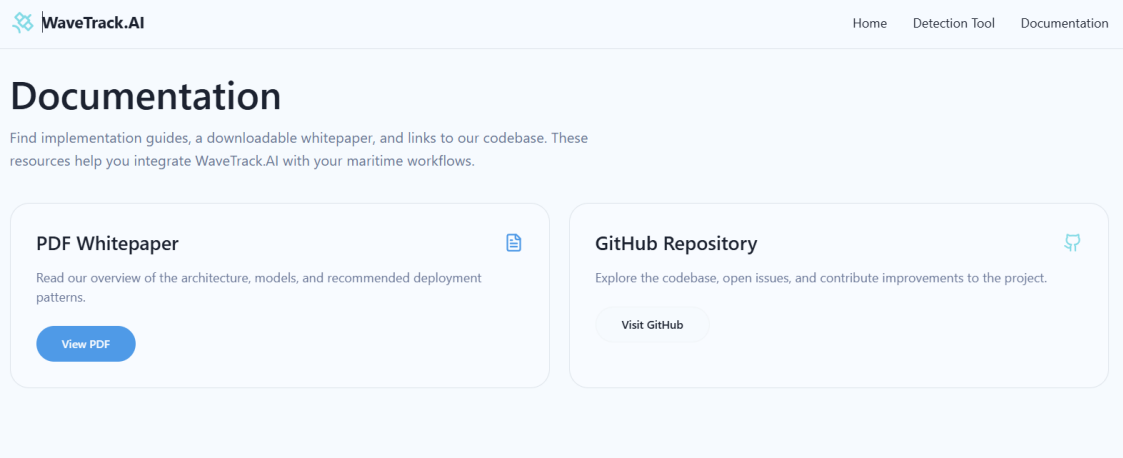


Figure 19: The documentation page with links to the whitepaper and GitHub.

13 Challenges and Limitations

Despite the advantages of using S-1 SAR data for maritime monitoring, several limitations and challenges must be acknowledged in the current version of our project.

1. Limited Temporal Resolution: A key limitation is the low temporal resolution of images. Sentinel-1A revisits the same area only once every 12 days, which introduces a bottleneck in the real time delivery of observations.

2. Noise in SAR Imagery: SAR images are inherently noisy. That makes preprocessing both essential and computationally intensive. This can reduce image clarity, increasing the risk of false positives or false negatives in detection.

3. Model heaviness: Power supply is a typical bottleneck for AI-products, which extend to this one. Even though this project implements YOLOv5-nano, which is a lean model, there is expected to be bottlenecks with high data volume.

4. Data Volume and Storage Constraints Sentinel-1 scenes, especially in full-resolution IW mode, are large (hundreds of MB per scene). For large-scale or continuous monitoring, data quickly accumulates into terabytes, requiring robust storage solutions and fast pipeline. Efficient data handling becomes a critical part of system design, especially when working in remote or bandwidth-limited environments.

14 Future Work

This section will outline the potential work that can be done to expand the scope of the project.

14.1 Velocity and direction estimation

A promising direction for extending this project is the estimation of ship velocity and direction from SAR Single Look Complex (SLC) data, without relying on AI-based models. Two complementary approaches can be combined: wake based kinematics and Doppler derived line-of-sight (LOS) velocity. Together they provide a full velocity vector and heading estimate for moving ships.

When a ship moves through the ocean, it generates a Kelvin wake pattern, visible in SAR imagery as a characteristic V-shape with a semi-angle of approximately $\sim 19,5^{o}$ in deep water. This can be seen in figure 20.

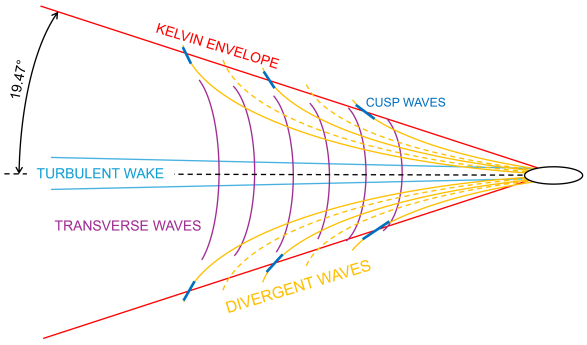


Figure 20: Illustration of a ship wake in deep waters.(Mazzeo et al., 2024)

The wake consists of transverse and divergent waves whose wavelength and orientation are directly related to the ship's speed. For deep water waves, the dispersion relation is (Zamparelli et al., 2020):

$$c_p(\lambda) = \sqrt{\frac{g\lambda}{2\pi}} \tag{5}$$

where c_P is the phase speed, λ is the wavelength, and g is the gravitational acceleration. If a wake wave crest propagates at an angle α to the ship's track, the ship's speed must satisfy the kinematic constraint:

$$v = \frac{c_p(\lambda)}{\sin\alpha} = \frac{1}{\sin\alpha}\sqrt{\frac{g\lambda}{2\pi}} \tag{6}$$

An important simplification occurs on the Kelvin cusp lines. Here the dominant wavelength λ_θ is directly related to the ship speed:

$$v = \sqrt{\frac{\sqrt{3}\, g\, \lambda_\theta}{4\pi}} \tag{7}$$

Thus, by detecting the wake arms and measuring their wavelengths, both heading (the bisector of the V-pattern) and speed magnitude can be estimated.

SLC data contain not only amplitude but also phase history, which allows estimation of a target's radial velocity. A moving target induces a shift in the Doppler centroid frequency, known as the Doppler Centroid Anomaly (DCA). After subtracting the expected (geometric) Doppler centroid from metadata, the residual Doppler frequency can be converted into LOS velocity (Zamparelli et al., 2020):

$$v_r = \frac{\lambda}{2} f_{\text{res}} \tag{8}$$

In practice, for SLC products such as Sentinel-1 IW, the Doppler centroid method is sufficient for estimating the sign of motion and for validating wake-based speed estimates.

By combining wake geometry with Doppler measurements, it is possible to reconstruct the full two-dimensional velocity vector of a ship. The wake provides the basis for determining both the heading and the magnitude of the speed. In cases where the Kelvin wake pattern is weak, the elongation axis of the ship itself in the SAR image can serve as a substitute. The actual speed can then be estimated from wake based wavelength analysis, either through the general dispersion relation or by applying the cusp-line shortcut equations.

While the wake analysis yields the course and speed magnitude, the direction of travel, whether the ship is moving forward or backward along this heading, cannot be resolved from wake geometry alone. This ambiguity can be resolved using the sign of the Doppler derived radial velocity, which indicates whether the ship is moving toward or away from the radar. Once both wake and Doppler information are available, a further consistency check can be performed. The relationship between the Doppler radial velocity v_r and the total speed v is given by

$$v = \frac{v_r}{\cos\beta}, \tag{9}$$

where β is the angle between the ship's heading and the radar's line of sight. If the radial velocity derived from Doppler is inconsistent with the wake-based estimate, the measurement can be flagged or refined accordingly (Mazzeo et al., 2024).

This fusion approach combines the strengths of both methods. The wake provides a physics-based estimate of the ship's magnitude and heading, while the Doppler anomaly contributes a sign and

validation layer, ensuring more robust and reliable velocity estimation.

To apply this method in practice, one must rely on the metadata included in Sentinel-1 SLC products, such as

- Radar wavelength λ
- Incidence angle $\imath$
- Doppler centroid f_{dc}
- Azimuth FM rate ψ
- PRF
- Orbit and attitude

These parameters provide the necessary context for both radiometric calibration and Doppler based velocity estimation.

References

(2025a). Copernicus sentinel-1 mission – wave mode. SentiWiki.

(2025b). Copernicus sentinel-1 products– level0 products. SentiWiki page describing Level-0 SAR products.

(2025). Synthetic aperture radar (sar) — earth observation data basics. Earth Observation Data Basics section on SAR.

Achiri, L., Guida, R., and Iervolino, P. (2018). Sar and ais fusion for maritime surveillance. In Proceedings of the IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI), pages –, Palermo, Italy.

Chen, J., Kao, S., He, H., Zhuo, W., Wen, S., Lee, C., and Chan, S. G. (2023a). Run, don't walk: Chasing higher flops for faster neural networks. `https://arxiv.org/abs/2303.03667v1`. arXiv preprint arXiv:2303.03667v1.

Chen, J., Kao, S., He, H., Zhuo, W., Wen, S., Lee, C.-H., and Chan, S.-H. G. (2023b). Run, Don't Walk: Chasing Higher FLOPS for Faster Neural Networks. arXiv preprint arXiv:2303.03667.

Del Prete, R., Graziano, M. D., and Renga, A. (2023). Unified framework for ship detection in multi-frequency sar images: A demonstration with cosmo-skymed, sentinel-1, and saocom data. Remote Sensing, 15(6):1582.

Grover, A., Kumar, S., and Kumar, A. (2018). Ship detection using sentinel-1 sar data. In ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, IV-5, pages 317–324, Dehradun, India. Proc. ISPRS TCV Mid-term Symposium "Geospatial Technology – Pixel to People" (20–23 Nov 2018).

Hou, Q., Zhou, D., and Feng, J. (2021). Coordinate Attention for Efficient Mobile Network Design. arXiv preprint arXiv:2103.02907.

Hu, C., Chen, H., Sun, X., and Ma, F. (2025). Polarimetric sar ship detection using context aggregation network enhanced by local and edge component characteristics. Remote Sensing, 17(4):568.

Iervolino, P., Guida, R., and Whittaker, P. (2015). A novel ship-detection technique for sentinel-1 sar data. In 2015 IEEE 5th Asia-Pacific Conference on Synthetic Aperture Radar (APSAR), pages 797–801.

Katsamenis, I., Karolou, E. E., Davradou, A., and Kalogeras, D. (2022). Tracon: A novel dataset for real-time traffic cones detection using deep learning. In Proceedings of the 2022 International Conference on Artificial Intelligence Applications and Innovations (AIAI). Springer. Figure 1: The architecture of the YOLOv5 model, which consists of three parts: (i) Backbone: CSPDarknet, (ii) Neck: PANet, and (iii) Head: YOLO Layer.

Mazzeo, A., Renga, A., and Graziano, M. D. (2024). A systematic review of ship wake detection methods in satellite imagery. Remote Sensing, 16(20):3775.

WaveTrack (2025). Detect your vessel from space. `https://preview--wavewatch-ai-portal.lovable.app/`. Accessed: 2025-08-18.

Wei, S., Zeng, X., Qu, Q., Wang, M., Su, H., and Shi, J. (2020). Hrsid: A high-resolution sar images dataset for ship detection and instance segmentation. IEEE Access, 8:120234–120254.

Wei, S., Zhang, H., Zeng, X., Zhou, Z., Shi, J., and Zhang, X. (2022). Carnet: An effective method for sar image interference suppression. International Journal of Applied Earth Observation and Geoinformation, 114:103019.

Wikipedia contributors (2025). Synthetic-aperture radar. Wikipedia.

Xu, X., Zhang, X., and Zhang, T. (2022). Lite-yolov5: A lightweight deep learning detector for on-board ship detection in large-scene sentinel-1 sar images. Remote Sensing, 14(4):1018.

Zamparelli, V., De Santi, F., Cucco, A., Zecchetto, S., De Carolis, G., and Fornaro, G. (2020). Surface currents derived from SAR doppler processing: An analysis over the naples coastal region in south italy. Journal of Marine Science and Engineering, 8(3):203.

Zhang, X., Liu, C., Yang, D., Song, T., Ye, Y., Li, K., and Song, Y. (2023). RFAConv: Innovating Spatial Attention and Standard Convolutional Operation. arXiv preprint arXiv:2304.03198.