Ship Detection in Sentinel-1 SAR Imagery Using Deep Learning

Project Report

(B.Tech. Computer Science and Engineering)

A Project work Report submitted to National Remote Sensing Center in fulfillment of internship at NRSC-ISRO

Submitted by

MEDHA PRASAD

Under the guidance of

Mr. Anil Yadav

Scientist 'E'

AS&CID, RSA

NRSC, ISRO Hyderabad

भारत सरकार
अन्तरिक्ष विभाग
राष्ट्रीय सुदूर संवेदन केन्द्र
बालानगर, हैदराबाद -500 037, तेलंगाना, भारत
टेलिफोन : +040-23879572-76
+040-23879261-65
फैक्स :+040-23878648

Government of India
Department of Space
National Remote Sensing Centre
Balanagar, Hyderabad - 500 037, Telangana, India
Telephone : +040-23879572-76
+040-23879261-65
Fax : +040-23878648

CERTIFICATE

Date: 28 June 2024

This is to certify that the project entitled **"Ship Detection in Sentinel-1 SAR Imagery Using Deep Learning"** is a bonafide work carried out by **Medha Prasad**, at National Remote Sensing Centre, Hyderabad during the period from **17 May 2024** to **28 June 2024** in offline mode under my virtual guidance. The student is pursuing **B.Tech** in **Computer science and Engineering** from **The International Institute of Information Technology - Hyderabad**.
She is very hardworking, sincere and intelligent and has completed the assigned task successfully.

Anil Yadav
Scientist Engineer, Application Software &
Computing Infrastructure Division, RSAA
National Remote Sensing Centre
Indian Space Research Organisation
Balanagar, Hyderabad, Telangana

DECLARATION

I hereby declare that the project report entitled “Ship Detection in Sentinel-1 SAR Imagery Using Deep Learning”, is an authentic work which has been carried out by me at “NRSC, ISRO Hyderabad” as part of my internship starting from May 2024 and ending in June 2024 under the supervision of my guide Mr. Anil Yadav.

I hereby affirm that the content presented in this report is original and the culmination of my individual efforts. Furthermore, it has not been previously submitted to any educational institution for the purpose of obtaining a degree.

Place: Hyderabad, Telangana

Date: 28/06/2024

MEDHA PRASAD

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to the National Remote Sensing Centre, ISRO, for offering me the opportunity to work on this project and for providing all the necessary facilities.

I extend my deepest appreciation to my project guide, Mr. Anil Yadav, Scientist 'E', Application Software & Computing Infrastructure Division (AS&CID), Remote Sensing Area (RSA) at the National Remote Sensing Centre (NRSC), Indian Space Research Organisation (ISRO), for his invaluable guidance, advice, and efforts in facilitating this project. I am truly thankful for his steadfast support throughout this study. His expertise and encouragement have been instrumental in my learning and development.

I would also like to extend my gratitude to Smt. Savitha Sunkari, Scientist 'E', Student Project Interface Division (SPID), Management System Area (MSA), for her support in setting up the requirements for initiating this internship. Additionally, I express my sincere appreciation to Dr. Jaya Saxena, Scientist 'F' and Head of SPID, MSA, for her continuous support and assistance during the internship.

This internship has been a valuable learning experience and has significantly contributed to my personal and professional growth. I am grateful for the support and guidance I received, and I look forward to applying these experiences in my future endeavors.

MEDHA PRASAD

ABSTRACT

This project focuses on developing deep learning models for ship detection in Sentinel-1 SAR imagery using the HRSID_JPG dataset. This dataset is a benchmark for evaluating ship detection and instance segmentation methodologies. The study explores a deep learning pipeline employing the U-net and YOLOv8 models. A novel preprocessing step was introduced to remove outliers such as island and shore bright spots from the imagery, leveraging recent advancements in segmentation methodologies tailored for SAR imagery. Techniques include automatic contrast stretching, threshold-based and area-based filtering using connected components rather than traditional pixel-based approaches. While exact quantification across multiple images pre and post-processing wasn't feasible, these methods are expected to enhance ship detection accuracy by addressing inherent challenges in SAR imagery. Evaluation of the YOLOv8 model, trained with a pre-trained YOLOv8n-seg model over 75 epochs with a batch size of 4, demonstrates promising results: Precision (P=0.923), Recall (R=0.705), mAP50 (0.831), and mAP50-95 (0.527). Post-preprocessing, YOLOv8 shows satisfactory performance on SAR imagery. Resource constraints limited thorough training of the U-net model. Future work could involve customizing YOLOv8 for specific ship detection needs and augmenting the HRSID dataset to further improve model performance. This report summarizes the project's objectives, methodologies, key findings, and outlines future research directions in satellite-based ship detection using deep learning.

Keywords: Ship detection, YOLOv8, U-Net, HRSID_JPG dataset, Preprocessing techniques, Connected components

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

Background

Ship detection from remote sensing imagery is pivotal for maritime security, encompassing traffic surveillance, combating illegal fisheries, monitoring oil discharges, and overseeing sea pollution. Traditional systems like automatic identification system (AIS), long-range identification and tracking (LRIT) and vessel monitoring system (VMS) primarily aid in collision avoidance and vessel tracking but fall short in monitoring non-cooperative ships. VMS, reliant on ship-installed components, provides continuous real-time vessel monitoring but lacks coverage for vessels without or with faulty systems, such as smaller fishing boats and passenger vessels exempt from regulations. Earth Observation through satellite imaging offers a comprehensive solution by detecting all vessel types, irrespective of onboard monitoring systems, making it invaluable for areas where traditional methods or VMS are impractical. Combining VMS with satellite imaging enhances effectiveness in maritime surveillance, crucial for monitoring illegal activities like oil spills and ensuring safety and sustainability in global industries such as fishing, shipping, and energy transport. Synthetic Aperture Radar (SAR) stands out as the leading remote sensing technique due to its capability to operate day and night, in all weather conditions, over large areas, making it indispensable for ship detection. ([1], [9])

SAR-based ship detection, despite its advantages, encounters challenges such as false alarms due to speckles and the reduced dimensions of the targets compared with the sensor spatial resolution. These complexities make automated interpretation of SAR images challenging, despite vessels being visually discernible. Manual visual inspection for ship detection in SAR images is labor-intensive, underscoring the necessity for automated detection methods. The Constant False Alarm Rate (CFAR) method, while widely used, faces difficulties in effectively mitigating false alarms. ([1], [9])

Deep learning models have emerged as promising solutions for SAR ship detection, leveraging advanced image processing techniques and large training datasets like the HRSID_JPG dataset. Models such as U-net and YOLOv8 are particularly effective in object detection and segmentation tasks, aiming to improve detection accuracy by addressing the challenges posed by SAR imagery.

This project aims to enhance ship detection accuracy in Sentinel-1 SAR imagery by developing novel preprocessing techniques to remove outliers and evaluate their impact on model performance. By advancing preprocessing methods and leveraging state-of-the-art deep learning models, this study contributes to the ongoing efforts in maritime surveillance, environmental monitoring, and other applications reliant on satellite-based ship detection.

Problem Statement

Ship detection in Sentinel-1 SAR imagery faces significant challenges due to high noise levels and interference from bright spots generated by land features like islands and shores. These issues severely impact the accuracy and reliability of ship detection, critical for applications in maritime surveillance and environmental monitoring. Current methods often struggle to effectively preprocess SAR imagery to mitigate these challenges, particularly in normalizing pixel values ranging from 0 to over 20,000 down to a standardized scale of 0 to 255. This preprocessing step is pivotal as it directly influences the performance of subsequent image segmentation models.

Traditional normalization techniques may inadequately handle bright outliers, such as those from shores or islands, potentially leading to reduced pixel values for ships which are in general, identified in the SAR imagery as very bright features because of the corner reflection. This diminishes the visibility of ships in the imagery and compromises the detection accuracy of deep learning models designed for ship detection in SAR data.

Main Objective

The main objective of this project is to develop preprocessing techniques to remove outliers in Sentinel-1 SAR imagery and to evaluate their impact on ship detection performance. This involves developing and implementing novel preprocessing steps, training deep learning models on the HRSID_JPG dataset, and exploring the effectiveness of different models, including U-net and YOLOv8, in improving ship detection accuracy.

Specific Objectives

The specific objectives of the study are:

- To explore and evaluate the effectiveness of the U-net and YOLOv8 models for ship detection in SAR imagery, with particular attention to their learning performance and usability.
- To research and evaluate various methods for preprocessing to remove outliers such as island and shore bright spots from the SAR imagery.
- To conduct a preliminary study on existing ship detection methodologies and their limitations.
- To design and implement a deep learning pipeline incorporating the chosen model and preprocessing techniques. This included:
    - Training the model on the HRSID_JPG dataset.
    - Evaluating the performance of the implemented models on the validation set of the HRSID_JPG dataset using key metrics.
    - Analysis of model predictions on preprocessed SAR images.

DATA SOURCES AND TOOLS USED

Data Sources

- **Alaska Satellite Facility (ASF)**: Provides high-quality Sentinel-1 SAR data, supporting Earth science research through comprehensive data services.
- **HRSID:** With the development of satellite technology, the latest SAR satellite imaging mode can provide higher resolution of SAR images, which is conducive to detecting ship targets [5]. High Resolution SAR Images Dataset (HRSID) is a dataset for ship detection, semantic segmentation, and instance segmentation tasks in high-resolution SAR images. The High-Resolution SAR Images Dataset contains 116 co-polarized and 20 cross-polarized SAR imageries. The original imageries for constructing HRSID are 99 Sentinel-1B imageries, 36 TerraSAR-X and 1 TanDEM-X images. As for dataset construction, under the overlapped ratio of 25%, 136 panoramic SAR imageries with ranging resolution from 1m to 5m are cropped to 800 x 800 pixels SAR images. There are 5604 cropped SAR images and 16951 ships in HRSID, and the creators have divided HRSID into a training set (65% SAR images) and test set (35% SAR images). The spatial resolution of SAR images is as follows: 0.5m, 1m, and 3m. The color depth of the images is 8 bits (one channel). The annotations of each SAR image constitute a .json file in the format of Microsoft Common Objects in Context (MS COCO) [2].

Tools Used

- **Colab**: Google's cloud-based platform for collaborative Python coding in Jupyter Notebooks, offering free access to GPU and TPU resources.
- **Google Drive**: Cloud storage service for secure file storage and access from any device, integrated with Google Colab for easy sharing and collaboration.
- **Python**: A versatile programming language known for simplicity and a rich ecosystem of libraries, ideal for web development, data analysis, AI, and scientific computing.
    - **NumPy**: For numerical operations and array manipulations.
    - **Pandas**: For data manipulation and analysis.
    - **Matplotlib**: For data visualization.
    - **Scikit-learn**: For machine learning algorithms.
    - **TensorFlow**: For deep learning models and includes tools for image preprocessing, model building, training, and evaluation.
        - Keras: High-level neural networks API running on top of TensorFlow, used for building and training deep learning models.
    - **OpenCV**: For image processing.
    - **GDAL**: For geospatial data manipulation, used to read, write, and process raster and vector geospatial data.
    - **Ultralytics**: Provides tools for training and deploying YOLO (You Only Look Once) models for object detection, including converting COCO annotations and training models with ease.
    - **skimage.morphology**: For advanced morphological operations on images, including max_tree for hierarchical segmentation and label for connected component labeling.
- **Jupyter Notebook**: An open-source web application for creating interactive documents with live code, visualizations, and narrative text, supporting various programming languages.

INPUT AND OUTPUT FILES

Input

1. **Sentinel-1 product images in ".tiff" format**: Sentinel-1, the first spacecraft in the European Space Agency's Copernicus Program, includes two satellites, Sentinel-1A and Sentinel-1B, orbiting in the same plane. These satellites are equipped with C-band Synthetic Aperture Radar (SAR), allowing them to collect data in any weather condition, day or night. Product types useful for ship detection are typically EW and IW in L1 GRD product type, with dual polarization.
   Description of the product I used:
   Mission: Sentinel-1A (S1A)
   Product type: Ground Range Detected (GRD) - provides detected imagery projected to the ground range.
   Beam mode: Interferometric Wide Swath
   Processing level: 1
   Resolution (Minimum separation the sensor can discriminate) class: H (high)
   Product class: S (SAR standard)
   Polarization: VV (Vertical Transmit, Vertical Receive) +VH (Vertical Transmit, Horizontal Receive)
   The specific products used for analysis include: (Data extracted from the measurement folder (VH polarization) within the ".SAFE" product. This folder contains binary data, provided in GeoTIFF format for Level-1 products)
   - South China Sea: S1A_IW_GRDH_1SDV_20240314T105535_20240314T105604_052977_0669B8_C321.SAFE
   - Bay of Bengal: S1A_IW_GRDH_1SDV_20240311T121225_20240311T121250_052934_06684D_7AF5.SAFE
   - Red Sea: S1A_IW_GRDH_1SDV_20240323T152959_20240323T153028_053111_066EE3_3DFF.SAFE

2. **HRSID** is an **open-source dataset** used for ship detection in Sentinel-1 SAR imagery. I have used the HRSID_JPG image dataset for my work due to storage constraints, although the high-fidelity SAR images in PNG format were available. The files I used were:
   - HRSID_JPG/JPEGImages: Contained all the 800 x 800 sized images.
   - HRSID_JPG/annotations/test2017.json: Included segmentation and related information for the test set.
   - HRSID_JPG/annotations/train2017.json: Included segmentation and related information for the training set.

   This dataset was used to train image segmentation models for ship detection.

Output

1. **Processed Image Chunks**: Preprocessed images from input TIFF file, divided into 800 x 800 sized .jpg images, are saved in the folder specified by OUTPUT_DIR_PROCESSED='/content/drive/My Drive/processed_divided_image_chunks/'.

2. **Direct Image Chunks**: Directly cut images (for the input file) without preprocessing are saved in the folder specified by OUTPUT_DIR_DIRECT='/content/drive/My Drive/divided_image_chunks/'.
3. **Runs Folder**: (These files are organized in this manner due to the functionality of the Ultralytics library)
    a. **Training results** are stored in the runs/segment/train directory.
        i. The best-performing model weights are saved as best.pt and the final model weights as last.pt in the weights subfolder.
    b. **Prediction Results**: If any image is given to the model for prediction, the results are stored in runs/segment/predict by the Ultralytics library. The code also plots the predicted mask.
    c. **Validation Results**: Validation results are stored in runs/segment/val.

PROJECT WORKFLOW

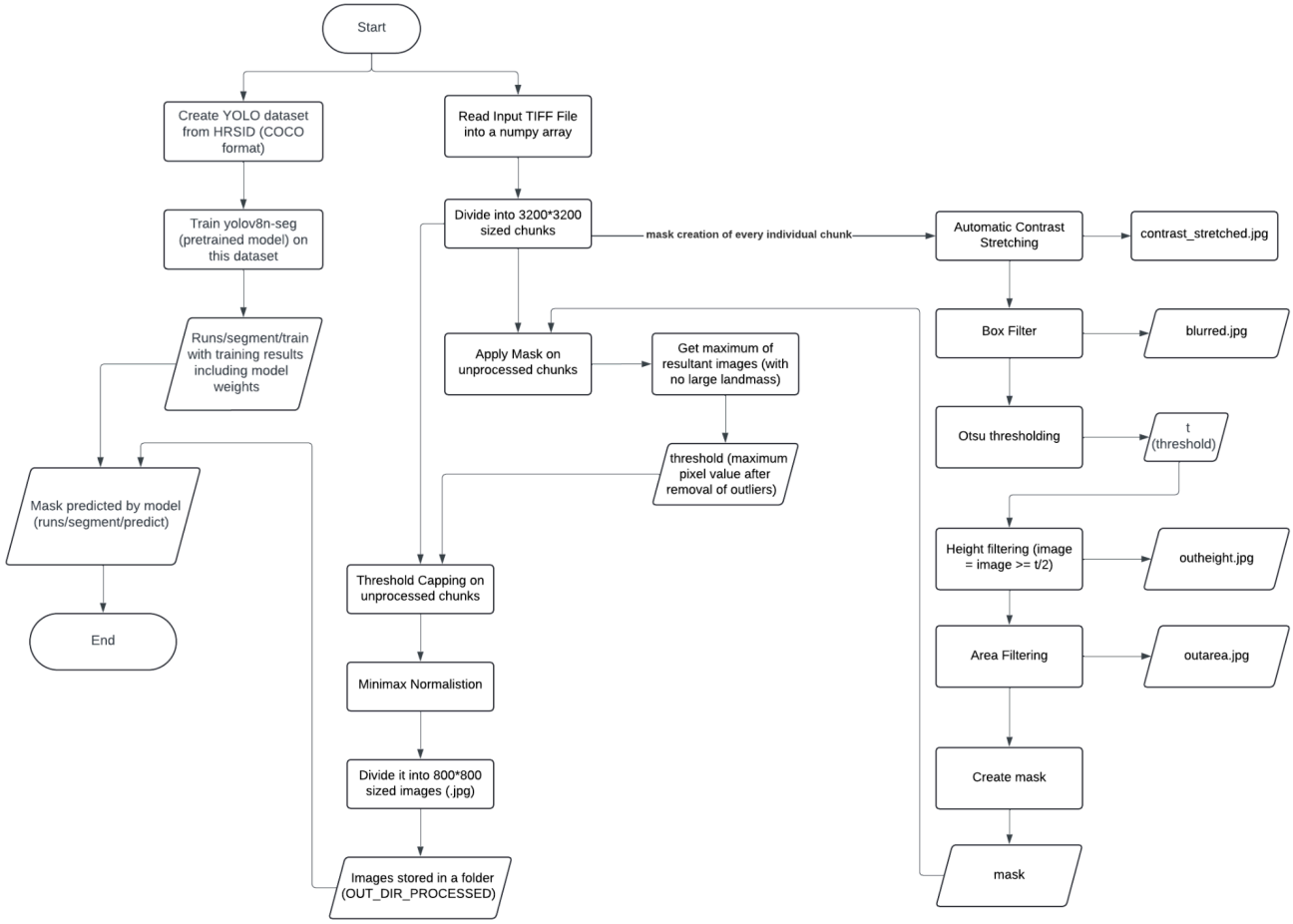


Fig. 1. Project Workflow

DATA PREPROCESSING METHODOLOGY

In the preprocessing stage of our ship detection project using Sentinel-1 SAR imagery, challenges when normalizing the pixel values, which can range from 0 to over 20,000, down to a scale of 0 to 255. This step is crucial for the subsequent segmentation process, as normalization affects the performance of the image segmentation models.

A simple normalization technique could result in diminished pixel values for ships, which are inherently bright due to their corner reflection [10], especially when the images contain outliers such as bright spots from shores or islands. These outliers can skew the normalization process, leading to lower intensity values for ships and consequently impairing the model's ability to detect them accurately.

To mitigate false alarms and enhance the detection accuracy, we implemented a preprocessing step inspired by a methodology described in a paper focused on ship detection in SAR imagery [1]. The paper highlights the inadequacy of classical segmentation techniques, such as standard thresholding and local thresholding, due to non-bimodal histograms and the broad range of gray-level values assumed by both ship targets and the sea background. Instead, the authors employed mathematical morphology and connected component transforms to pre-screen potential ship pixels.

Central to their approach are connected operators, which interact with images through connected components. These operators preserve image contours while simplifying by removing components that would erode entirely, contrasting with operations at the pixel level. For ship detection, the methodology focuses on distinguishing ship pixels from other segments, reducing false alarms from islands and other features. Filtering criteria are based on height and area, initially extracting bright targets based on height and subsequently filtering based on area.

In adapting this approach, our preprocessing pipeline segments large landmasses and eliminates them from consideration, leaving potential ship areas for further analysis. By capping the brightness threshold to the maximum value identified in these potential ship areas, our normalization step effectively disregards outliers, optimizing the data for subsequent processing stages, particularly deep learning models.

Methodology

This adaptation tailors the methodology to specifically address the challenge of reducing false alarms from bright spots on shores and islands, preparing the data robustly for deep learning-based ship detection models rather than techniques such as radon or wavelet transforms. This approach involves the following steps:

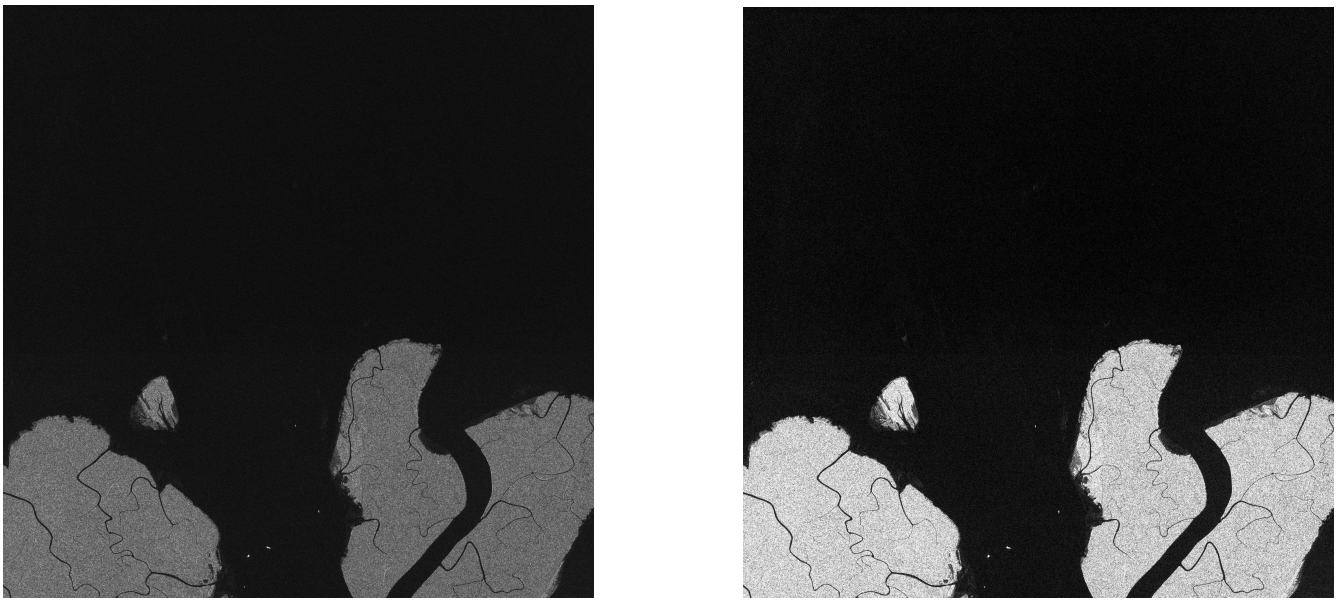
1. The input TIFF file is initially converted into a numpy array.
2. To optimize processing time and improve the handling of local information in SAR imagery [1], the array is segmented into 3200x3200 sized chunks. While the method in [1] used 3000x3000 pixel chunks, I adopted 3200x3200 pixel chunks to ensure that after processing, segments of 800x800 pixels were achieved. Using slightly larger initial chunks aids in accurately identifying landmasses during area filtering.
3. For each segment, the preprocessing includes creating a mask to isolate potential ship areas from the rest of the image:
    3.1. **Automatic Contrast Stretching**: This technique improves contrast by stretching the range of intensity values to span the full range allowed by the image type (e.g., 0 to 255 for 8-bit images). Unlike histogram equalization, which applies a nonlinear scaling, contrast stretching applies a linear scaling, resulting in a less harsh enhancement [3].
        - I have taken the 5th and 95th percentiles of the histogram image in the histogram are taken as the lower (c) and upper (d) bounds (that is, 5% of the pixel in the histogram will have values lower than *c*, and 95% of the pixels will have values higher than *d*).

This approach mitigates the influence of outliers as compared to taking the lowest and highest pixel values currently present in the image as c and d.

$$P_{out} = (P_{in} - c)\left(\frac{b-a}{d-c}\right) + a$$

- Each pixel *P* is scaled using the following function: (a = 0, b = 255)
- Values below 0 are clipped to 0 and values about 255 are clipped to 255.

Example of application:The example I am taking was part of a satellite image taken near west bengal shore bay of bengal this example taken from red sea



(a) *(b)*

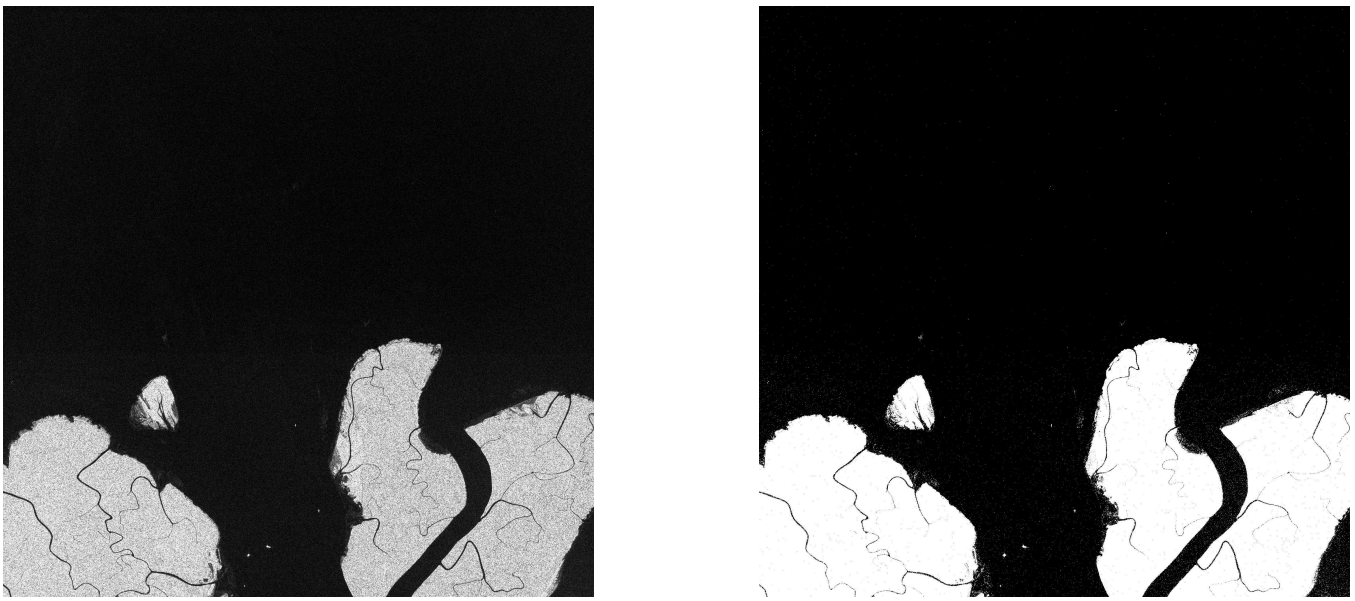
Fig.2. (a) Image before contrast stretching (b) Image after contrast stretching

3.2. **Box Filter**: Apply box filter with a kernel size of (10,10) to enhance the connectivity of larger land features within the image. This blurring step ensures that larger landmasses become more connected, facilitating easier identification and removal in subsequent steps.

3.3. **Thresholding**: Otsu's thresholding method is employed to automatically determine a threshold (t) for separating potential landmasses from the sea based on brightness. Otsu's method calculates an optimal threshold by minimizing the intra-class variance of pixel intensities in the image histogram, effectively dividing the image into foreground (landmasses) and background (sea) regions.

3.4. **Height Filtering**: Following Otsu's thresholding, the height threshold is computed as half of the calculated threshold value (t/2). This adjustment aims to selectively capture lower intensity values, primarily representing landmasses, while effectively filtering out sea areas, typically characterized by near-zero intensities.

The resultant binary image, referred to as `outheight`, is generated by comparing each pixel in the blurred image to the height threshold. Pixels with intensities greater than or equal to t/2 are set to 255 (white), indicating potential landmasses, while pixels below this threshold are set to 0 (black), representing other areas.



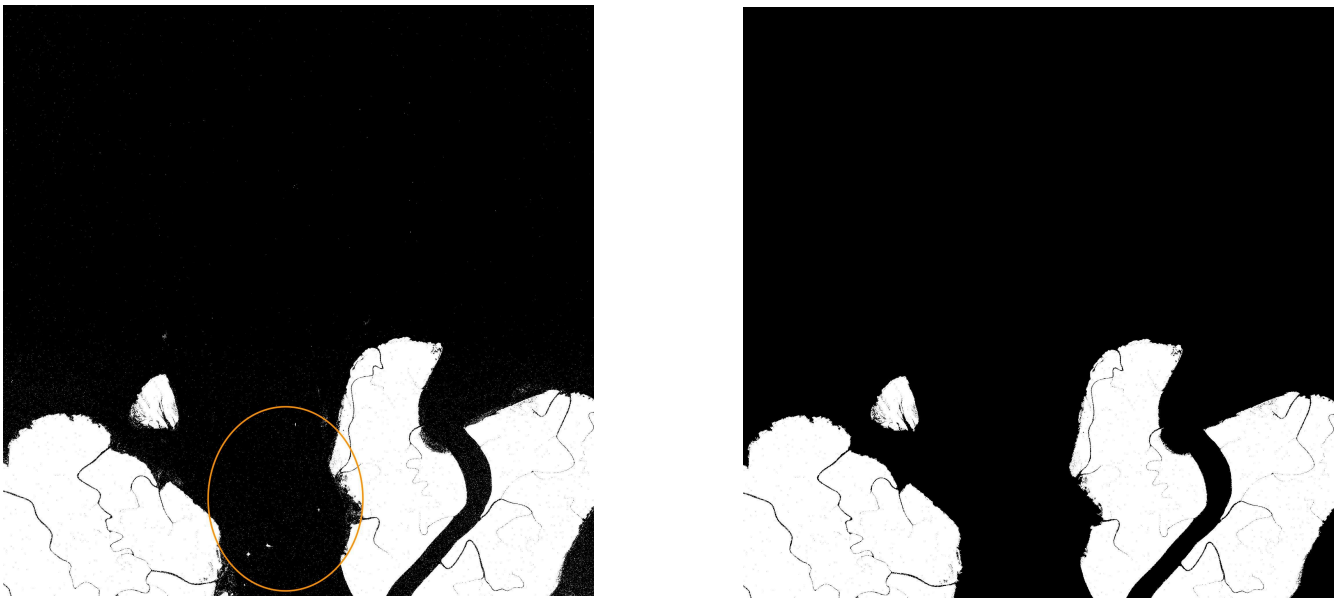
(a) *(b)*

Fig.3. (a) Blurred Image (b) Image after height filtering (outheight)

3.5. **Area-Based Filtering**: In image processing, connected components are groups of adjacent pixels that share the same pixel intensity value. Identifying these components allows us to analyze and process distinct objects within the image. This step involves:

3.5.1. Using the `label()` function, connected components in the binary image are identified and labeled, and the total number of features (connected components) detected. The function can consider either 4-connectivity (only vertical and horizontal neighbors) or 8-connectivity (vertical, horizontal, and diagonal neighbors). `connectivity=2` is specified to ensure that diagonal connections are considered.

3.5.2. Components smaller than the area threshold (3200 pixels) are removed from the image, creating a filtered binary image (`outarea`). 3200 is an approximate number chosen hoping to eliminate potential ship pixels.

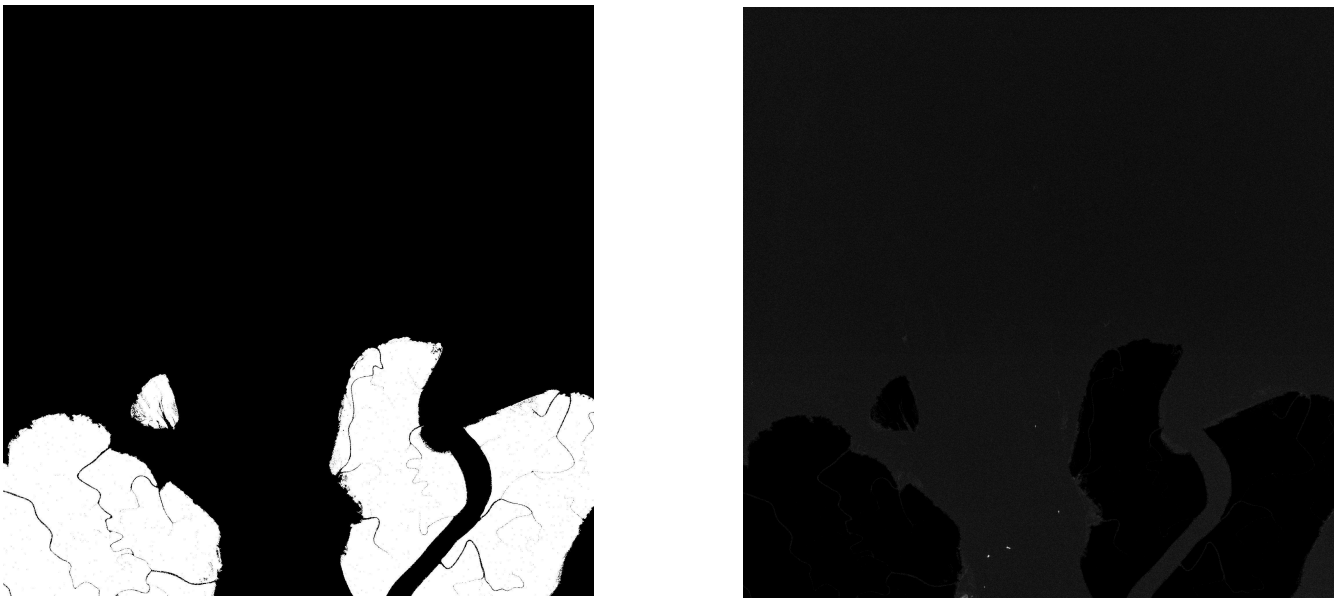


(a) *(b)*

Fig.4. (a) outheight (b) Image after area filter (outarea)

3.6. **Mask Creation**:A mask is created where `outarea` has a value of 0, indicating regions that are not large landmasses (`mask = outarea == 0`).

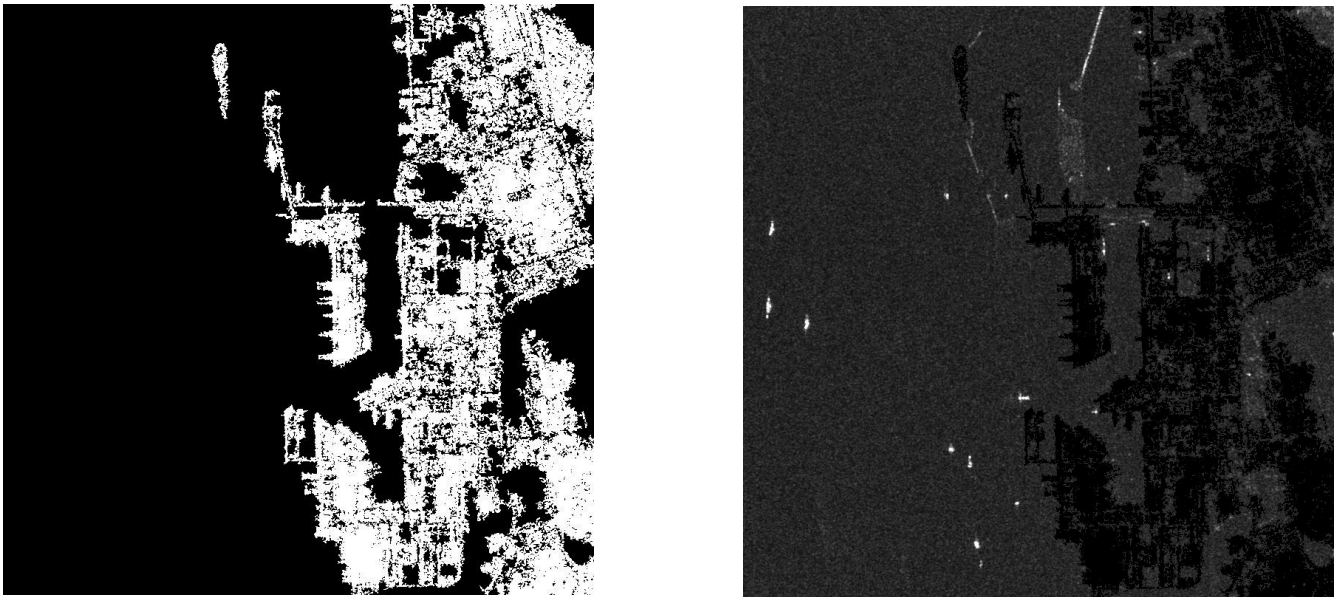
4. The mask is applied to the initial image segment to retain original values from the image where the mask is true. The maximum pixel value in the `resultant_image` (i.e., the brightest pixel value in the smaller components) is identified as the threshold.



(a) *(b)*

Fig.5. (a) outarea (b) mask created

Clearly the part of the image with land masses has been removed. Another example:



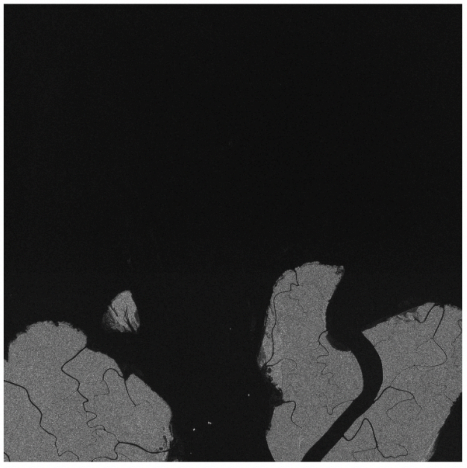
(a) *(b)*

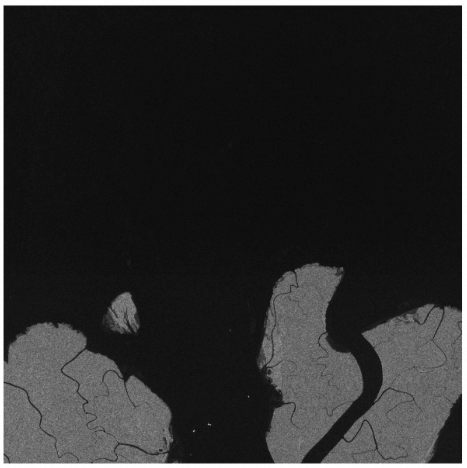
Fig.6. (a) outarea (b) mask created

5. **Cap Values at Threshold**: The original image segment values are capped at the identified threshold, ensuring that any value greater than the threshold is set to the threshold.

   For instance, in the first image, where the maximum pixel value corresponded to a ship, the maximum value remained unchanged after the capping process. Conversely, in the second example, the maximum pixel value was reduced by 21% after the thresholding, indicating a successful elimination of outlier bright spots from non-ship objects.

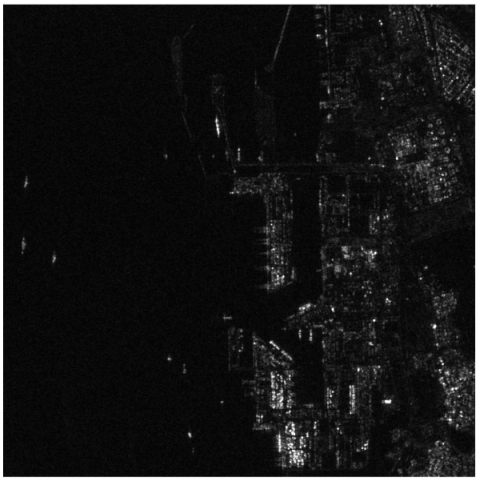
6. The capped image is normalized so that the minimum value becomes 0 and the maximum value becomes 255. The normalized image data type is changed to `uint8.` Final images for these two examples:





(a) *(b)*

Fig.7. (a) unprocessed initial image (b) final processed image (no change)



(a) *(b)*

Fig.8. (a) unprocessed initial image (b) final processed image

7. The 3200x3200 processed image segments are then divided into 800x800 tiles. These 800x800 tiles are stored in JPEG format.

Example Regions shown above

Bay of Bengal, near West Bengal Shore: The first example was part of a satellite image taken near the West Bengal shore in the Bay of Bengal.

Red Sea: The second example was from the Red Sea.

Limitations, Recommendations and Conclusion

These preprocessing steps collectively ensure that the normalization process does not diminish the intensity values of ships, thereby improving the performance of our image segmentation model in detecting ships accurately while minimizing false alarms.

To further enhance these techniques, they can be refined and extensively tested on satellite imagery encompassing varied sea conditions and regions, which cover the diverse navigation scenarios of ships at sea. While exact quantification across multiple images pre and post-processing wasn't feasible within the project period, these methods are expected to enhance ship detection accuracy by addressing inherent challenges in SAR imagery. Continued exploration and refinement of these preprocessing methods will be crucial for advancing the robustness and reliability of ship detection models in satellite-based applications.

U-Net

U-net is a convolutional neural network designed for image segmentation [6], characterized by its U-shaped structure. The network combines high-resolution spatial information from the contracting path with high-level semantic information from the expansive path, making it effective in capturing context and achieving precise localization. The architecture includes encoder-decoder paths with skip connections, allowing the model to extract and refine features for accurate segmentation. This makes U-net particularly suitable for tasks like ship detection in SAR imagery, where spatial accuracy is crucial.

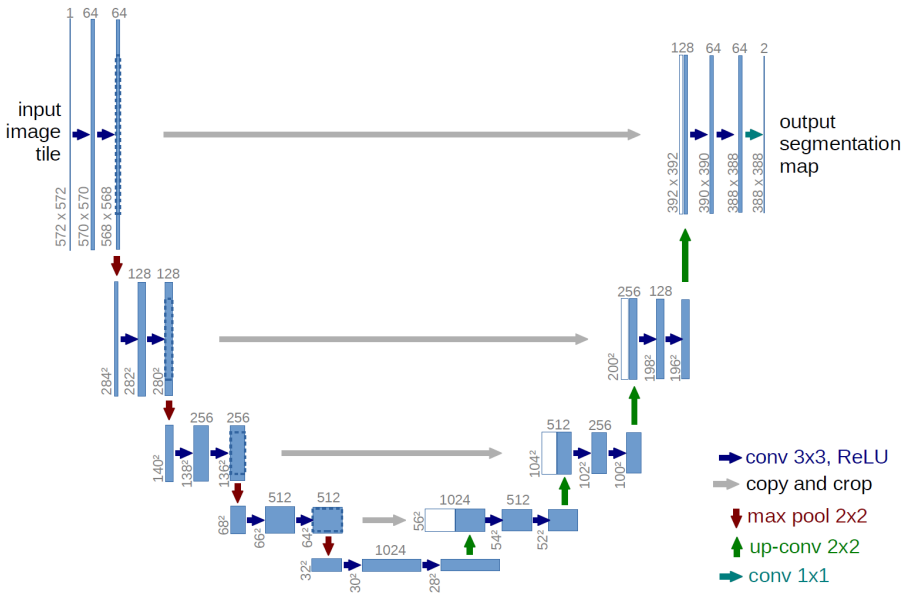
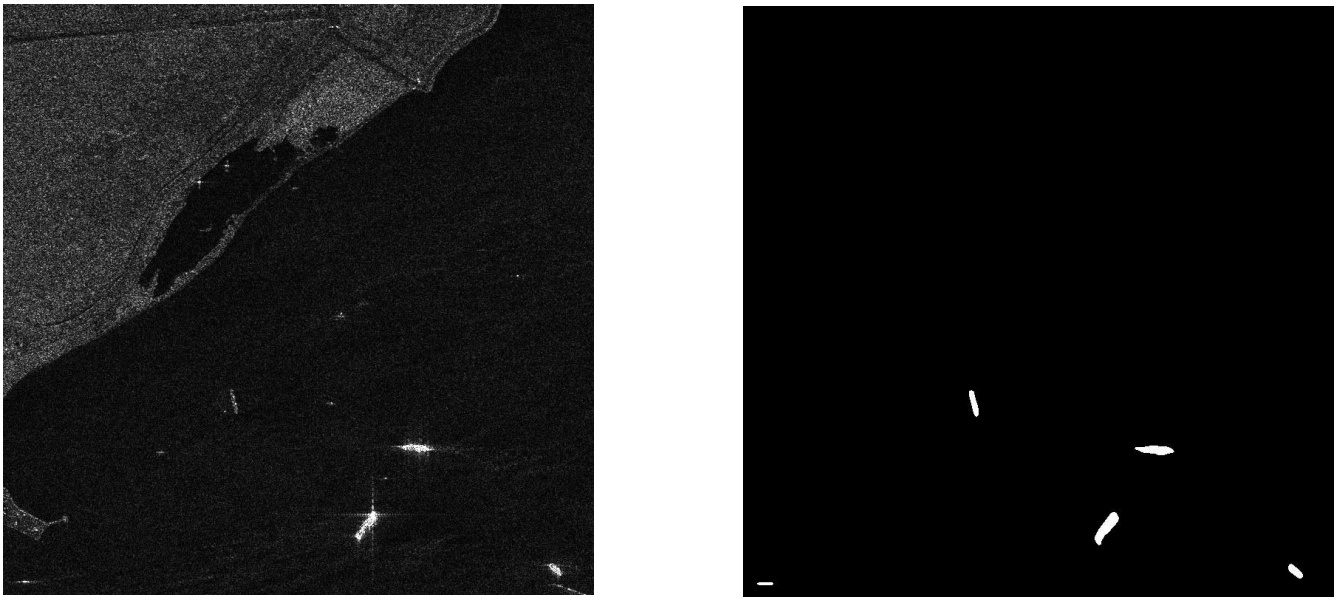


Fig. 9. U-Net Architecture [6]

Dataset Preparation

The dataset used for this project is the HRSID_JPG dataset. The following steps were taken to create the dataset for training the U-net model:

1. **Annotation Transformation:** A script was used to generate binary masks (target) from the annotations (JSON format). The masks were created by drawing polygons defined in the annotation files onto blank images, resulting in binary masks where ships are represented by pixel value 1 and the background by pixel value 0. For eg:



(a) *(b)*

Fig.10. (a) image (b) target mask

2. **Directory Structure:** The dataset was organized into directories to facilitate easy loading using TensorFlow's ImageDataGenerator:

```
Data/
train_images/
            train/
                img1, img2, img3, ......

train_masks/
            train/
                msk1, msk2, msk3, ......

val_images/
            val/
                img1, img2, img3, ......
```

```
val_masks/
            val/
```

3. **Image and Mask Generators:** Separate data generators for images and masks were set up to read from the respective directories and apply data augmentation. The training images and masks were resized from 800x800 to 256x256 to fit into system RAM.

Model Training Pipeline

The training process for the U-net model involved several key steps to ensure efficient and accurate learning from the HRSID_JPG dataset:

1. **Model Preparation:** The U-net model was built using a series of encoder and decoder blocks, each consisting of convolutional layers, batch normalization, and activation functions. The encoder captures context, while the decoder refines spatial details.
2. **Compilation:** The model was compiled with the Adam optimizer and binary cross-entropy loss. Key metrics such as precision and recall were used to monitor performance during training.
3. **Training Configuration:** The images and masks were resized to 256x256 to fit into system RAM. Data generators were used to read and augment the images and masks from their respective directories, facilitating efficient loading and preprocessing.
4. **Training Execution:** The model was trained on the HRSID_JPG dataset using the training and validation generators. A batch size of 4 was used due to resource constraints.
5. **Model Evaluation and Saving:** After training for a set number of epochs, the model's performance was evaluated on the validation set. Finally, the trained model was saved for future inference and analysis.

Limitations, Recommendations and Conclusion

The U-net model was initially trained from scratch with the aim of evaluating its effectiveness in ship detection in Sentinel-1 SAR imagery. However, due to time and storage constraints, the model could only be trained on a batch size of 4 with resized images. Despite these limitations, the U-net model demonstrated its potential by being able to process and segment ships in the SAR imagery.

Given the constraints, the U-net model was only trained for 5 epochs. During this period, other models, such as YOLOv8, were also explored. It was found that YOLOv8, especially when fine-tuned from a pre-trained YOLOv8n-seg model, demonstrated superior performance early in the training process. As the primary goal was to build and test a model for evaluating the preprocessing techniques, YOLOv8 was adapted as the main model for this project.

The decision to pivot to YOLOv8 was based on its ability to quickly achieve high performance, making it more suitable given the project's resource constraints. Future work could involve further training and optimization of the U-net model with more resources to fully explore its potential in this application.

YOLOv8

Developed by Ultralytics, the creators behind YOLOv3 (PyTorch fork) and YOLOv5, YOLOv8 represents the latest evolution in this series of models.It represents a cutting-edge advancement in real-time object detection and image segmentation models. Built on state-of-the-art deep learning techniques, YOLOv8 excels in both speed and accuracy, making it versatile for a wide range of applications across different hardware platforms, from edge devices to cloud APIs [4].

As of late 2023, YOLO remains a benchmark in real-time object detection, recognized for its ability to deliver robust performance across various datasets and scenarios. YOLO (You Only Look Once) architecture is known for its efficiency in real-time object detection. YOLOv8 builds upon the success of its predecessors, leveraging advancements in convolutional neural networks to predict bounding boxes and class probabilities swiftly and efficiently in a single pass through the network.

This relatively new model architecture offers built-in support for not only object detection, but also **instance segmentation** and **image classification** tasks. The model itself was constructed in PyTorch, and is **capable of running on both CPUs and GPUs**. This versatility is accessible through a Python package and a command-line interface, facilitating ease of use and integration into diverse workflows.

YOLOv8 includes a range of pre-trained segment models tailored for specific tasks (segmentation, classification and detection). The pretrained YOLOv8 models are trained on extensive datasets like COCO (Common Objects in Context), empowering them to recognize and classify a wide array of objects effectively. Pretrained models in YOLOv8 enable efficient fine-tuning on specific datasets or tasks, minimizing the necessity for lengthy training processes and accelerating deployment. This approach harnesses learned patterns for rapid adaptation, enhancing accuracy and speed in object detection applications across diverse environments.

Due to resource constraints, I utilized YOLOv8n-seg (n is for ‘nano’) for this project, the base model optimized for minimal computational cost and efficient performance. This variant is ideal for applications prioritizing speed or operating under limited hardware resources.

Dataset Preparation

The dataset, formatted in MS COCO (HRSID), is already organized into separate train and test sets. Using the provided JSON annotations, images were copied to corresponding directories (`data/images/train` and `data/images/val`) from a source directory (`JPEGImages`).

Annotations were converted from COCO format to YOLO format (`data/labels/train/img1.txt`, `data/labels/val/img1.txt`) using a custom conversion script, ensuring compatibility with YOLOv8. The final directory is organized as follows:

```
Data/
    images/
        train/
            P0082_2400_3200_1800_2600.jpg
            P0082_2400_3200_3600_4400.jpg
            ...
        val/
            P0054_1800_2600_2400_3200.jpg
            P0054_1800_2600_4200_5000.jpg
            ...
    labels/
        train/
            P0082_2400_3200_1800_2600.txt
            P0082_2400_3200_3600_4400.txt
            ...
        val/
            P0054_1800_2600_2400_3200.txt
            P0054_1800_2600_4200_5000.txt
            ...
```

Fig. 11. final directory structure

A `config.yaml` file was created to specify dataset paths and parameters for YOLOv8 model training.

```yaml
path: /content/gdrive/My Drive/ImageSegmentationYOLOv8/data
train: images/train
val: images/val

nc: 1
names: ['ship']
```

Fig. 12. Content of the `config.yaml` file

This setup streamlines the dataset preparation and configuration necessary for training YOLOv8 for ship detection.

Training the Model

Model: YOLOv8n-seg (pre-trained)

Training Parameters:

- **data:** `<path-to-config.yaml>`
- **imgsz:** 800
- **seed:** 42 (to ensure reproducibility)
- **batch:** 4 (due to resource constraints, the batch size is small)
- **workers:** 4
- **patience:** 90

The training results get stored in `runs/segment/train`

Evaluation Metrics

Key metrics tracked by YOLOv8 are precision, recall, mAP50 and mAP50-95.

Precision: The proportion of true positive predictions among all positive predictions made by the model. It measures how accurate the positive predictions are.

Recall: The proportion of true positive predictions among all actual positive instances in the dataset. It measures the model's ability to correctly identify positive instances.

mAP50: Mean Average Precision at IoU threshold of 0.5. It calculates the average precision of the model across different classes, averaged over different IoU thresholds, with a primary threshold of 0.5.

mAP50-95: Mean Average Precision averaged over IoU thresholds from 0.5 to 0.95. It provides a broader view of the model's precision across a range of IoU thresholds.

F1-score: The harmonic mean of precision and recall. It balances both precision and recall, providing a single metric that indicates the model's overall performance, especially useful when dealing with imbalanced datasets.

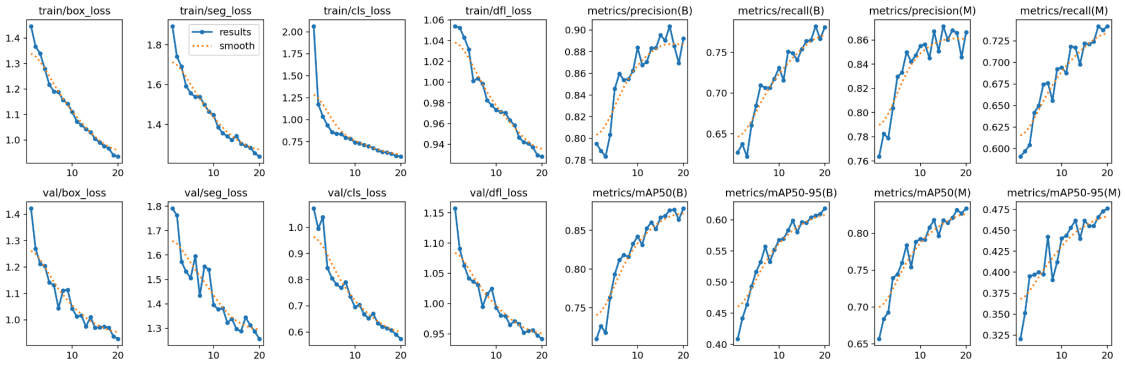
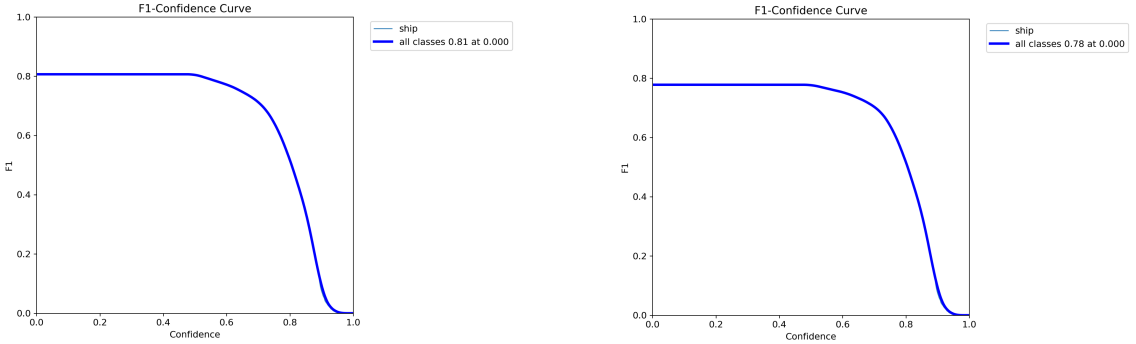


Fig. 13. The loss and evaluation metrics over the first 20 epochs

The above plot shows the initial convergence trend of the model. Here, the B and M stand for Box and Mask. For instance segmentation, you have metrics for both bounding boxes and masks as an instance segmentation provides both.

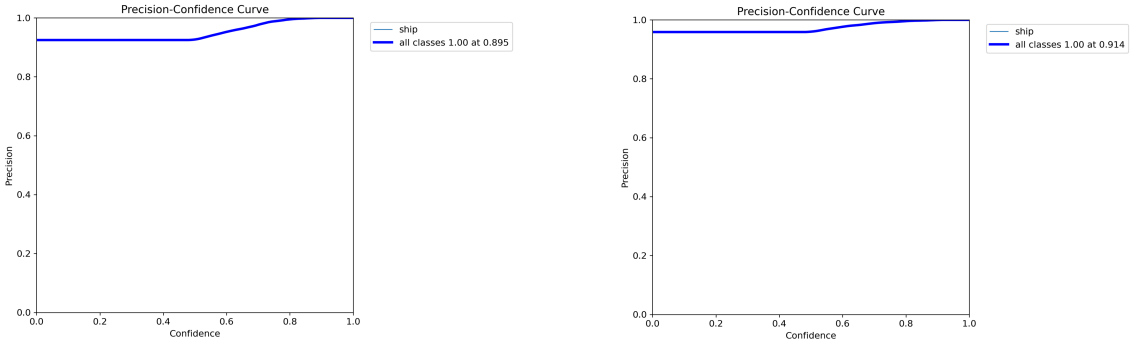
Even though the model had not yet fully converged, training was stopped at 75 epochs due to time constraints. However, the following results and plots provide insight into the model's promising results and performance.

Confidence Threshold Analysis:



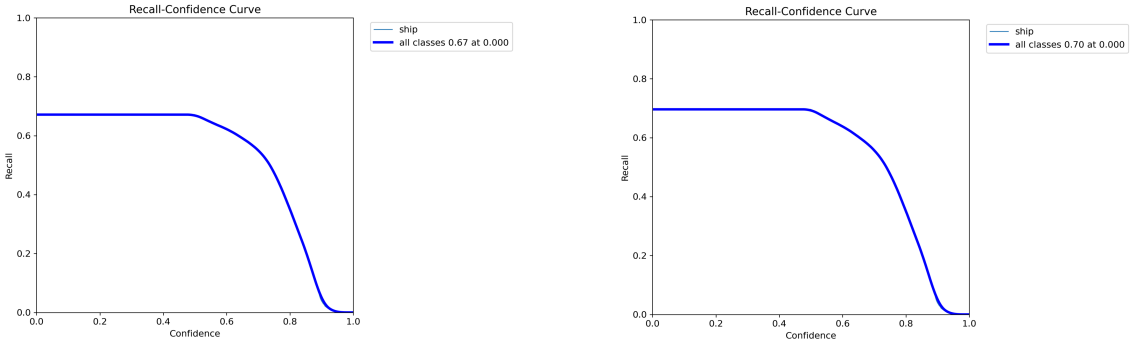
(a) *(b)*

Fig.14. F1 Score vs. Confidence (a) For Mask Prediction (b) For Bounding Box Prediction



(a) *(b)*

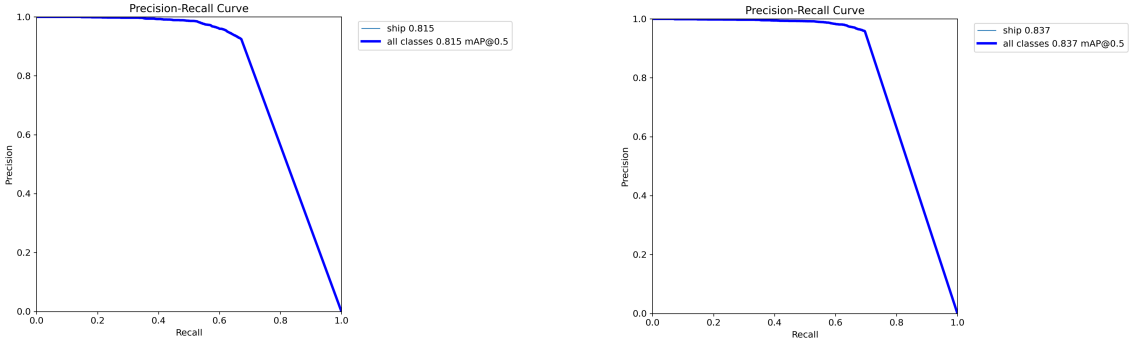
Fig.15. Precision vs. Confidence (a) For Mask Prediction (b) For Bounding Box Prediction



(a) *(b)*

Fig.16. Recall vs. Confidence (a) For Mask Prediction (b) For Bounding Box Prediction

These plots help in determining the optimal confidence threshold for validation (ships detected with a confidence lesser than that are not considered), balancing precision and recall to optimize the F1 score. A precision-recall curve is a plot that shows the trade-off between precision and recall for different threshold settings.



(a) *(b)*

Fig.17. Precision vs. Recall (a) For Mask Prediction (b) For Bounding Box Prediction

Validation of the Model

After training, the model was evaluated on the validation data (1962 images) with a chosen confidence threshold to ensure reasonable performance metrics.

Model Evaluation Parameters:

- **Model Path:** `<path-to-last.pt>` or `<path-to-best.pt>`
- **Validation Data:** `<path-to-config.yaml>`
- **Image Size:** 800
- **Batch Size:** 4
- **Confidence Threshold:** 0.56
- **IoU Threshold:** 0.6

Evaluation Results: (on various performance metrics)

For Bounding Box Prediction (B):

- **Precision (P):** `0.967`
- **Recall (R):** `0.739`
- **mAP50:** `0.86`
- **mAP50-95:** `0.683`

For Mask Prediction (M):

- **Precision (P):** `0.923`
- **Recall (R):** `0.705`
- **mAP50:** `0.831`
- **mAP50-95:** `0.527`

The validation results are stored in `runs/segment/val`.

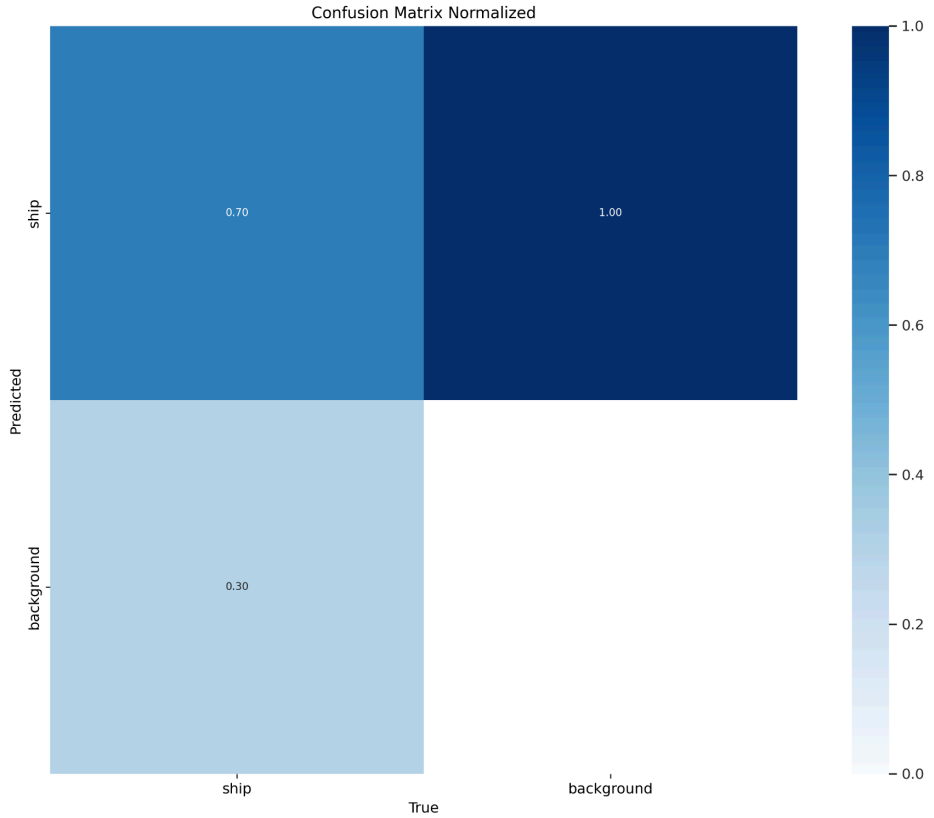
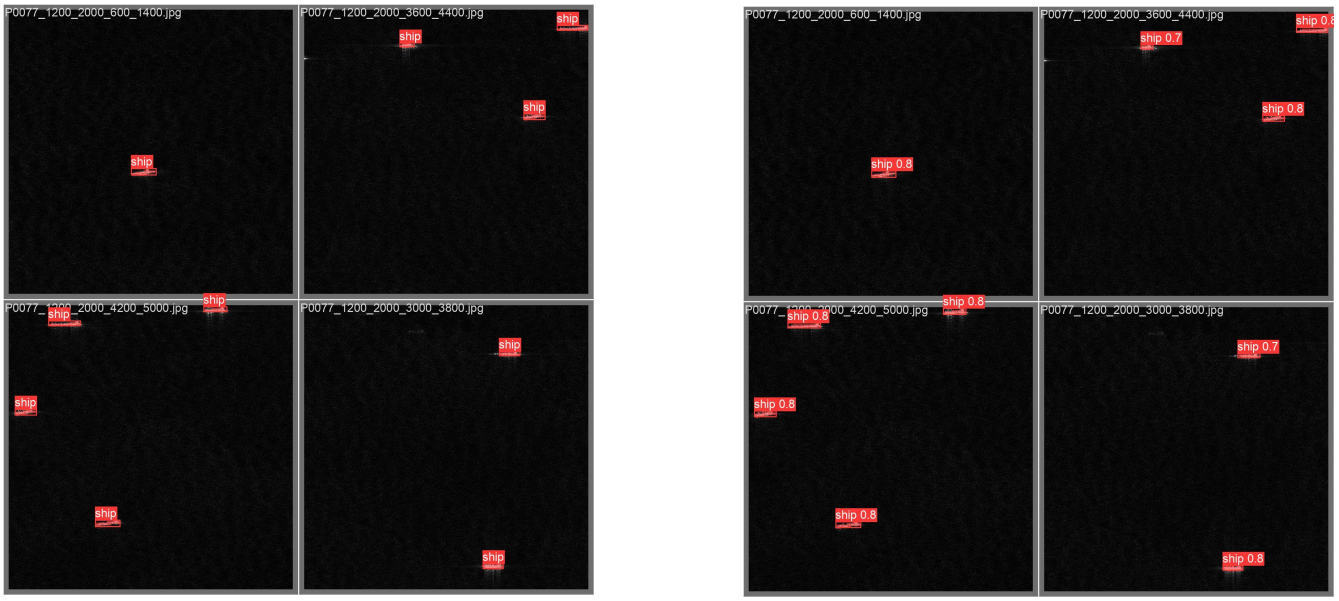


Fig. 18. Confusion Matrix Normalised



(a) *(b)*

Fig.19. Example of prediction (a) val_batch1_labels.jpg (b) val_batch1_pred.jpg (the number shows the confidence)

These metrics and plots demonstrate the model's ability to accurately identify and segment ships in the validation dataset, with high precision and a reasonable balance between precision and recall.

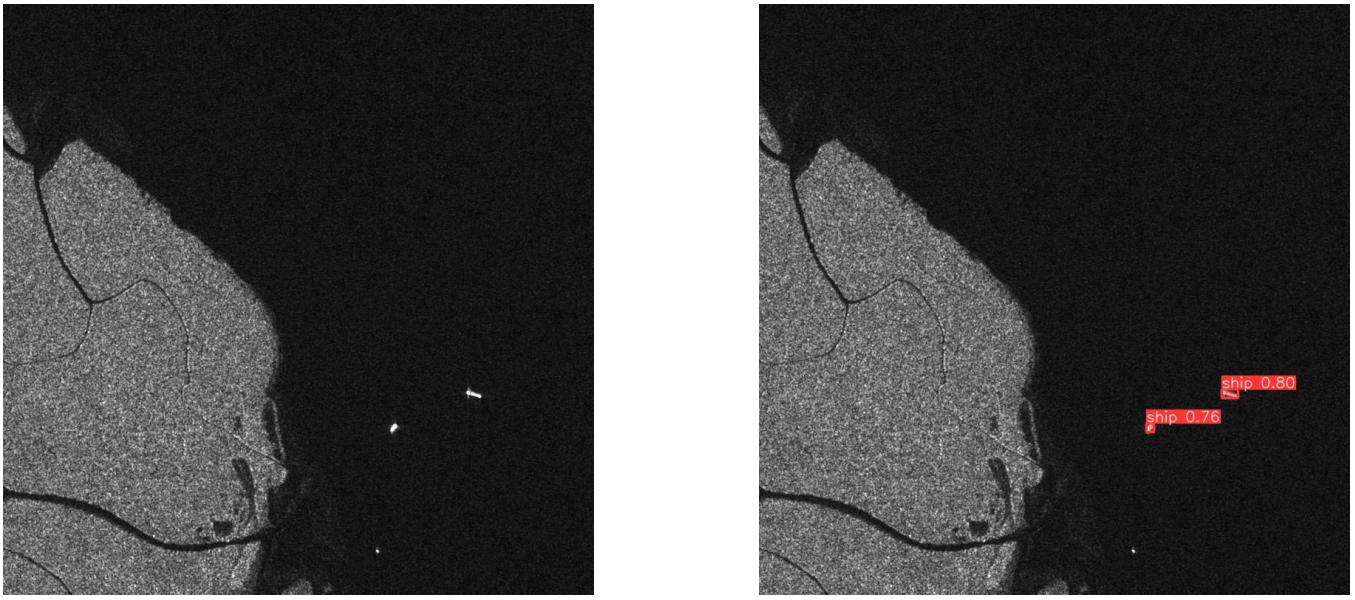
Prediction by Model for an Input Image

Finally, the model is used to predict masks for processed image tiles that it has not seen before. These are the tiles created during the data preprocessing step discussed earlier.

Model Prediction Parameters:

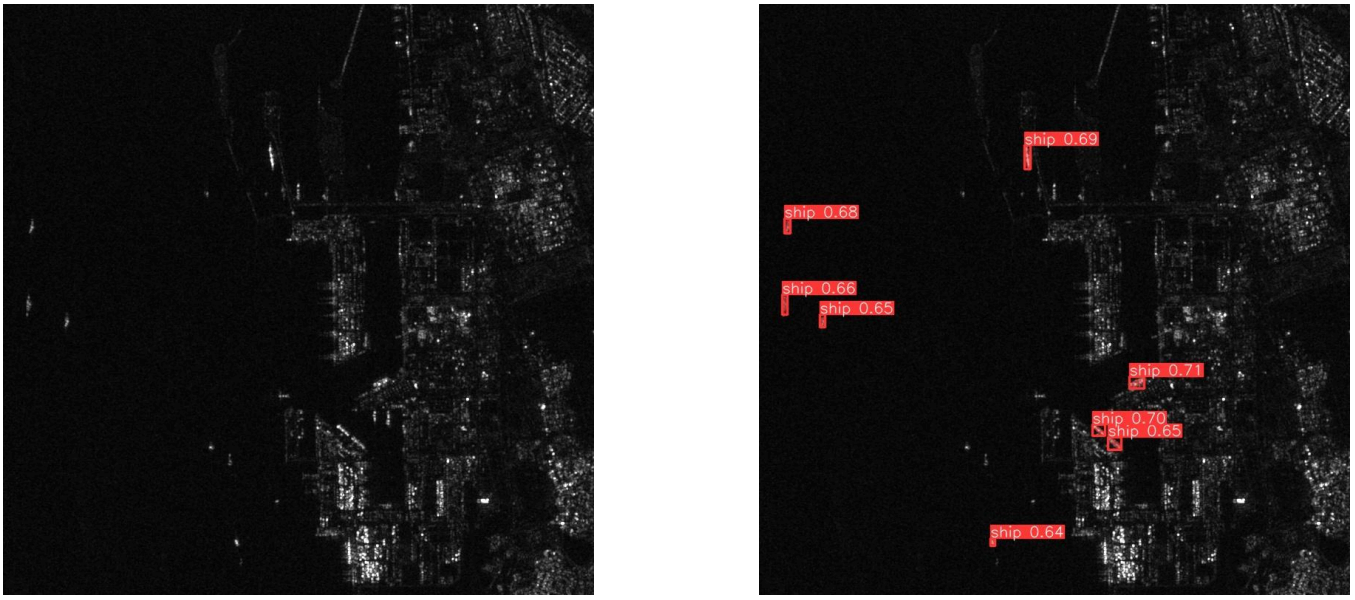
- **Model Path:** `<path-to-last.pt>` or `<path-to-best.pt>`
- **Image Path:** `<image-path>`
- **Image Size:** 800
- **Confidence Threshold:** 0.56

The predicted mask is plotted and stored in `runs/segment/predict`. Below are some examples of the detection on different images, demonstrating the model's performance:



(a) *(b)*

Fig.20. ***Example 1*** *(a) Input image (b) Model prediction*



(a) *(b)*

Fig.21. ***Example 2*** *(a) Input image (b) Model prediction*

As we can see, the model performs well in detecting and segmenting ships in the images, indicating the effectiveness of the pipeline in real-world scenarios.

Conclusion, Limitations and Recommendations

The model demonstrated promising results with high precision and reasonable recall, indicating its potential for effective ship detection in Sentinel-1 SAR imagery. Overall, the model performed well in both training and validation phases. However, the training was limited to 75 epochs, which was not sufficient for the model to fully converge. Moreover, the batch size was set to 4 due to resource constraints. Future endeavors could extend the training period of the model and increase the batch size with a hardware upgrade. Implementing data augmentation techniques to address class imbalance and improve model generalization would also be beneficial. While using a pre-trained model was efficient in minimizing the need for lengthy training processes and accelerating deployment, building a custom YOLO model from scratch could further fine-tune hyperparameters specific to ship detection, such as loss functions and optimizers, potentially leading to improved results. In summary, while the model has shown promising results, there are several avenues for improvement that could be explored to further enhance its performance.

CONCLUSION

Overall, the project focuses on developing robust ship detection models using Sentinel-1 SAR imagery, addressing inherent challenges in satellite-based applications. Innovative preprocessing techniques were implemented to optimize the normalization process crucial for accurate segmentation. By leveraging methodologies like mathematical morphology and connected component analysis, outliers were effectively filtered out, significantly improving detection accuracy.

Evaluation of deep learning models, particularly the U-net and YOLOv8, highlighted YOLOv8's superior performance under project constraints. Fine-tuned from a pre-trained model, YOLOv8 demonstrated high precision and reasonable recall, indicating its suitability for real-world applications despite resource limitations.

Moving forward, further refinement of these models with extended training periods, increased batch sizes, and enhanced data augmentation could enhance performance and robustness. Continued exploration of advanced preprocessing methods tailored to diverse sea conditions will be pivotal in advancing ship detection capabilities in satellite imagery.

In conclusion, while promising results have been achieved, ongoing research and development are essential to fully exploit the potential of deep learning in satellite-based ship detection, ensuring reliable performance across varying maritime environments.

SOURCE CODE

sar_data_preprocessing.ipynb

train_yolov8n_seg_custom_dataset.ipynb

u_net_model.ipynb

REFERENCES

1. Corbane, C., Najman, L., & Pecoul, E. (December 2010). A complete processing chain for ship detection using optical satellite imagery. *ResearchGate*. Retrieved from https://www.researchgate.net/publication/228346316_A_complete_processing_chain_for_ship_detection_using_optical_satellite_imagery
2. Shunjun Wei ; Xiangfeng Zeng ; Qizhe Qu ; Mou Wang ; Hao Su ; Jun Shi. HRSID: A High-Resolution SAR Images Dataset for Ship Detection and Instance Segmentation . IEEE Access
3. Fisher, R., Perkins, S., Walker, A., Wolfart, E., Brown, N., Cammas, N., Fitzgibbon, A., Horne, S., Koryllos, K., Murdoch, A., Robertson, J., Sharman, T., Strachan, C. (2004). Stretching: Histogram Stretching. *HIPR2*. Retrieved from https://homepages.inf.ed.ac.uk/rbf/HIPR2/stretch.htm
4. @software{yolov8_ultralytics, author = {Glenn Jocher and Ayush Chaurasia and Jing Qiu}, title = {Ultralytics YOLOv8}, version = {8.0.0}, year = {2023}, url = {https://github.com/ultralytics/ultralytics}, orcid = {0000-0001-5950-6979, 0000-0002-7603-6750, 0000-0003-3783-7069}, license = {AGPL-3.0} }
5. Zheng, Y., Er, M. J., Yi, G., & Shen, S. (July 2021). RepUNet: A Fast Image Semantic Segmentation Model Based on Convolutional Reparameterization of Ship Satellite Images. In *2021 6th International Conference on Automation, Control and Robotics Engineering (CACRE)* (pp. 1-3). DOI: 10.1109/CACRE52464.2021.9501337. Retrieved from https://www.researchgate.net/publication/353809581_RepUNetA_Fast_Image_Semantic_Segmentation_Model_Based_on_Convolutional_Reparameterization_of_Ship_Satellite_Images
6. Ronneberger, O., Fischer, P., & Brox, T. (May 2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv:1505.04597 [cs.CV]*. Retrieved from https://arxiv.org/abs/1505.04597
7. Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. *scikit-image: Image processing in Python*. PeerJ 2:e453 (2014) https://doi.org/10.7717/peerj.453
8. Copernicus. (n.d.). Sentinel-1. Sentiwiki. Retrieved July 1, 2024, from https://sentiwiki.copernicus.eu/web/sentinel-1
9. Bezerra, D.X., Lorenzzetti, J.A., & Paes, R.L. (2023). Marine Environmental Impact on CFAR Ship Detection as Measured by Wave Age in SAR Images. *Remote Sensing, 15*(13), 3441. https://doi.org/10.3390/rs15133441
10. Jiang, Q., Wang, S., Ziou, D., El-Zaart, A., Rey, M.T., Benie, G.B., & Henschel, M. (1998). Ship Detection in Radarsat SAR Imagery. *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics (Volume 5)*. https://doi.org/10.1109/ICSMC.1998.727570

For the Sentinel-1 SAR imagery:

- Primary Data: Copernicus Sentinel data 2024, processed by ESA.
- Modified Data: Data courtesy of ESA, ASF DAAC 2024, contains modified Copernicus Sentinel data 2024, processed by ESA.