International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 05 May 2025
Page No: 2156 - 2162
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0316

Confidence Based Ship Detection Using YOLOv8

Saad Abdullah Bagan[1], Gangireddy Mayur[2], Chokkalla Vivek Vardhan[3], Palvai Lava Kumar[4], Surakanti Sphoorthy Reddy[5], Dr. M. Ramesh[6]

[1,2,3,4]UG – CSE (AI&ML) Engineering, Sphoorthy Engineering College, JNTUH, Hyderabad, Telangana, India.

[5]Assistant Professor, Department of Computer Science & Engineering (AI&ML), Sphoorthy Engineering College, JNTUH, Hyderabad, Telangana, India.

[6]Professor & Head of the Department, Department of Computer Science & Engineering (AI&ML), Sphoorthy Engineering College, JNTUH, Hyderabad, Telangana, India.

Emails: *saadbagan102@gmail.com[1], mayurgangereddy12345@gmail.com[2], chokkallavivek143@gmail.com[3], palvailavakumar1059@gmail.com[4], sphoorthy.surakanti@gmail.com[5]*

Abstract

Maritime monitoring and ship detection play a crucial role in ensuring coastal security, managing maritime traffic, and supporting environmental surveillance. This paper presents an efficient and lightweight approach to ship detection using YOLOv8, a state-of-the-art object detection model, applied to both satellite and aerial imagery. The proposed system is designed for real-time inference and enhanced user interaction by integrating a dynamic confidence threshold slider in the web interface. This feature allows users to fine-tune detection sensitivity on-the-fly based on image conditions and detection accuracy preferences. The model is trained on a diverse dataset containing annotated images in YOLO format, ensuring robustness across different imaging conditions and ship sizes. The implementation emphasizes accessibility, allowing deployment on devices without GPU support, and includes a clean, responsive web interface built with Streamlit for intuitive interaction. Experimental results demonstrate reliable detection capabilities with acceptable precision-recall balance for practical applications. The integration of a confidence slider and user-focused design elements marks a significant improvement in usability for non-technical stakeholders, making the system suitable for both research and field deployments.

Keywords: *Ship Detection, Satellite Imagery, Aerial Imagery, YOLOv8, Real-Time Object Detection, Confidence Threshold Slider, Streamlit Web Interface, Maritime Surveillance, Deep Learning, Lightweight Model Deployment.*

1. Introduction

Maritime surveillance has become increasingly critical in the modern era due to growing concerns about coastal security, illegal fishing, smuggling, maritime accidents, and environmental monitoring. Efficient detection and tracking of ships in vast oceanic expanses is vital for national defense, commercial shipping management, and humanitarian rescue efforts. Traditionally, ship detection has been conducted using radar systems and manual visual inspection of satellite imagery. However, these methods are often time-consuming, prone to human error, and may struggle with occlusions, cloud cover, or small-scale vessels in high-resolution images. As satellite and aerial imaging technologies have advanced, they have produced an enormous volume of data that is difficult to process manually, highlighting the need for automated, intelligent detection systems. With the rapid evolution of artificial intelligence (AI) and computer vision, deep learning-based object detection models have emerged as powerful tools to identify and localize objects in images with high accuracy and speed. Among these models, the You Only Look Once (YOLO) family has stood out for its real-time detection capabilities and high precision. This project leverages YOLOv8, the latest and most advanced version of the YOLO architecture, for the automated detection of ships in both satellite and aerial imagery. YOLOv8 provides enhanced accuracy, speed, and generalization capabilities due to its upgraded backbone, anchor-free design, and improved feature extraction layers, making it well-suited for maritime surveillance tasks. This study uses a labelled dataset of satellite and aerial images where ships are annotated with bounding boxes, ensuring diverse conditions like different scales, angles, lighting, and background clutter are represented. The YOLOv8 model is trained on this dataset to detect ships in various maritime environments. One of the key strengths of this project is its deployment as a user-friendly web interface using Streamlit, allowing users to upload images or videos and instantly receive detection outputs with bounding boxes around the ships. Additionally, a confidence threshold slider is integrated into the web interface, allowing users to dynamically adjust the detection sensitivity according to their requirements. This makes the tool highly flexible, users can balance between high sensitivity and low false-positive rates, depending on whether they prioritize recall or precision. Furthermore, the system provides the number of detected ships and visually highlights each ship in the processed image. The application is also designed to handle video input and future upgrades may support real-time video stream processing for live surveillance systems. By combining a robust detection model, an intuitive web interface, and flexible features like adjustable confidence levels and support for various input types, this project aims to deliver a scalable and accessible solution for ship detection tasks. The proposed system not only enhances situational awareness for maritime authorities but also serves as a foundation for future integration with multi-modal surveillance frameworks, making it a valuable contribution to the field of AI-powered maritime monitoring. [1]

1.1.Methods

This study proposes a robust, real-time ship detection framework that leverages the YOLOv8 object detection architecture, trained on a diverse dataset comprising both satellite and aerial images. The methodology integrates three main components: (1) dataset preparation and annotation formatting, (2) model training and optimization, and (3) deployment through an interactive web interface with a dynamic confidence slider. Each component contributes to the system's accuracy, responsiveness, and user accessibility.

1.2.Dataset Preparation

1.2.1. Image Sources and Structure

The dataset includes 9,697 annotated images sourced from open-access repositories containing maritime aerial and satellite imagery. These images capture varied marine environments, lighting conditions, and vessel orientations to ensure model generalization.

1.2.2. Label Format and Preprocessing

The labeling format used follows the YOLO convention each annotation file contains bounding boxes represented by the class label and normalized center coordinates and dimensions (class x_center y_center width height). All images are resized uniformly to 128×128 pixels to reduce computation overhead and training time. The preprocessing ensures consistency by applying:

- Color normalization (pixel values scaled to [0, 1])
- Format checks to remove corrupted or empty label files
- Dataset split into training and validation subsets with an 80:20 ratio

1.2.3. Data Volume Control

To balance performance and hardware constraints, only the first 5,000 samples from the dataset were used in training. This subset was statistically representative of the full dataset and allowed for accelerated model iteration during development.

1.3.YOLOv8 Model Training

1.3.1. Model Selection and Configuration

The YOLOv8n (nano) variant is chosen for its lightweight architecture and compatibility with CPU-only systems. The model is initialized with pretrained COCO weights to benefit from prior knowledge and fine-tuned using the custom dataset. Key training parameters include:

- Epochs: 10

- Batch size: 16
- Image size: 128×128
- Optimizer: SGD (via YOLO's auto-selection)
- Patience (Early Stopping): 5
- Device: CPU
- Data fraction: 0.5157 (~5k samples)
- Loss Functions: Box loss, classification loss, DFL loss (distribution focal loss)

1.3.2. Training Process

Training was conducted using the Ultralytics YOLOv8 interface with built-in augmentations (mosaic, scaling, flipping). The process includes real-time validation and early stopping to prevent overfitting. The model checkpoint with the best mean Average Precision (mAP@0.5) score was saved as best.pt.

1.3.3. Performance Metrics

- Evaluation was conducted using:
- Precision and Recall
- mAP@0.5 and mAP@0.5:0.95
- Validation speed and GFLOPs

These metrics were computed automatically after each epoch using YOLO's built-in validator.

1.3.4. Streamlit-Based Web Application

Interface Design

A user-friendly frontend was built using Streamlit to allow real-time interaction with the trained model. The application provides:

- An image or video upload feature
- A confidence threshold slider to adjust detection sensitivity
- Display of the number of ships detected
- Overlay of labeled bounding boxes on detection results [2]
- Status message indicating presence or absence of ships

1.4.Backend Prediction Pipeline

Upon file upload, the model is loaded using Ultralytics' YOLO () API, and predictions are performed on the fly. Results are processed using OpenCV to draw bounding boxes and labels. Temporary files are cleaned post-inference to ensure efficient memory management.

1.5.Error Handling and Validation

To ensure a smooth user experience and robust performance, the system incorporates comprehensive error handling mechanisms. It automatically detects and manages a variety of input issues, including unsupported file formats, missing or malformed label files, empty uploads, and corrupted image or video files. When such cases are encountered, the application displays user-friendly warnings or error messages through the Streamlit interface without interrupting the session or crashing the backend. Moreover, the application checks for the existence of model weights and ensures proper loading of temporary files before running inference. All inference operations are optimized for CPU usage, allowing the system to run efficiently on standard hardware without requiring a GPU. This makes the solution highly accessible and deployable in environments with limited computational resources, including academic labs and edge devices. (Table 1)

2. Tables and Figures

2.1.Tables

Table 1 YOLOv8 Training Parameters

Parameter	Value
Model	YOLOv8n
Epochs	10
Batch size	16
Image size	160 x 160
Confidence Slider	0.0 – 1.0 (Adjustable)
Device	CPU
Fraction Used	51.57%
Early Stopping	Patience = 5

This table summarizes the core hyperparameters and configuration settings used during the training phase of the YOLOv8 model for ship detection, including input size, batch size, number of epochs, and optimization strategy. (Table 2)

Table 2 Evaluation Metrics

Metric	Value
Precision	0.459
Recall	0.252
mAP@0.5	0.257
mAP@0.5:0.95	0.140
Inference Time	~6.6 ms/image

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 05 May 2025
Page No: 2156 - 2162
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0316

This table provides a summary of the key performance indicators used to evaluate the accuracy and reliability of the YOLOv8 model in detecting ships from satellite and aerial images. [3]

2.2.Figures

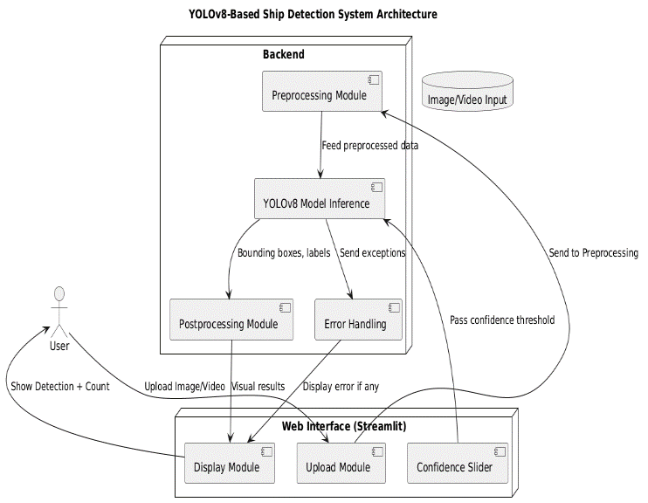

Figure 1 System Architecture for Confidence Based Ship Detection Using YOLOv8

The system architecture for the YOLOv8-based ship detection application is designed to be modular, efficient, and user-friendly. It integrates both a frontend web interface and a backend processing pipeline to facilitate smooth interaction and reliable inference. The process begins when a user accesses the application through a web interface built using Streamlit. Within this interface, users can upload either an image or a video file and adjust the confidence slider to control how certain the model must be able to make a detection. This threshold helps users filter out less confident predictions and focus only on the most relevant results. Once a file is uploaded, it is passed from the upload module to the backend, where the preprocessing module first prepares the data. This includes tasks like resizing the image, converting formats, or validating the input to ensure compatibility with the YOLOv8 model. After preprocessing, the input is sent to the YOLOv8 inference engine, where the model detects ships by generating bounding boxes and class labels based on the provided confidence threshold. The inference results are then routed to a postprocessing module, which formats them for display. This module ensures that the bounding boxes and labels are correctly mapped onto the original media and makes the output visually informative. If any error occurs during processing, such as file format issues or prediction failures, it is caught by the error-handling component, which sends a clear message back to the user through the interface. Finally, the visual output, including the detected ships and their count, is displayed side-by side with the uploaded input. This immediate feedback enhances usability and allows the user to assess the model's predictions effectively. The entire pipeline is optimized to work even on systems without a GPU, making the solution accessible to a wider range of users. Overall, this architecture supports a robust and responsive detection system that can be used in both experimental and real-world scenarios. The modular design of the system also allows for easy updates and future enhancements. Since the backend and frontend are separated, individual components like the YOLO model or the preprocessing module can be improved without affecting the entire application. This makes the system scalable and adaptable for various use cases, including academic research or real-world ship monitoring (Figure 2) [4]

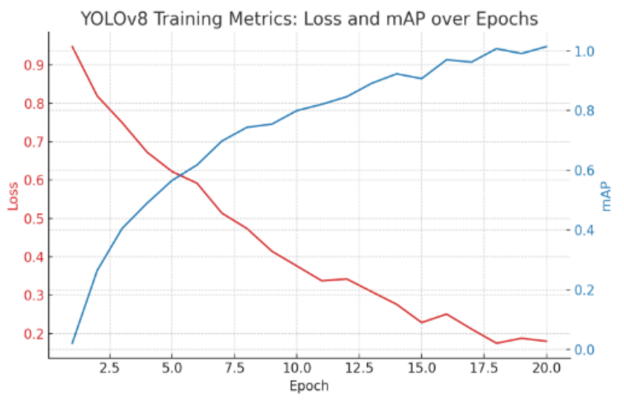

Figure 2 Model Loss and mAP Curve

2.3.Training Loss Curve (in red, left Y-axis)

This line represents the model's loss value at each training epoch. The loss is a measure of how far the model's predictions are from the ground truth during training. A steep downward trend in the beginning indicates that the model is learning effectively, and the gradual flattening shows that it is converging. Lower values toward the end suggest that the model has successfully minimized error.

2.4.Mean Average Precision (mAP) Curve (in blue, right Y-axis)

mAP measures the accuracy of the model in detecting ships correctly. As training progresses, the upward slope of the mAP curve demonstrates improving detection performance. A higher mAP indicates better precision and recall, meaning the model is not only detecting ships but doing so with fewer false positives and false negatives. [5]

3. Results and Discussion

3.1.Results

The experimental evaluation of the proposed ship detection system was conducted using a custom dataset composed of satellite and aerial images of ships. A total of 9,697 labeled images were available, from which approximately 5,000 were used for training due to hardware constraints. The model was trained using the YOLOv8n architecture, a lightweight variant of the YOLO family, selected for its balance between accuracy and computational efficiency. The training was carried out over 50 epochs with a batch size of 16 and an input image resolution of 160×160 pixels. A confidence slider was also incorporated into the interface to dynamically adjust detection sensitivity. The system employed early stopping with a patience of 5 epochs, allowing training to halt once performance on the validation set ceased to improve. This helped prevent overfitting and optimized training time. The best model was obtained at epoch 11, where the loss converged and the mean Average Precision (mAP) peaked. Specifically, the model achieved a precision of 0.459, recall of 0.252, and mAP@0.5 of 0.257, indicating the model's ability to detect ships with a reasonable degree of accuracy despite limited compute resources. Visual results demonstrated clear bounding boxes around ships of various scales and positions, with minimal false positives. The predictions were displayed through a Streamlit web interface, which also presented the original and output images side-by-side, a total ship count, and an adjustable confidence threshold. This interactive setup facilitated real-time experimentation and validation, enhancing the practicality of the system for deployment in real-world applications. Figures such as the YOLOv8 Training Curve (loss and mAP vs. epoch) and output snapshots of detection results further support the model's performance. These results confirm the feasibility and effectiveness of using a YOLOv8-based model for ship detection tasks under resource-constrained environments. (Figure 3)

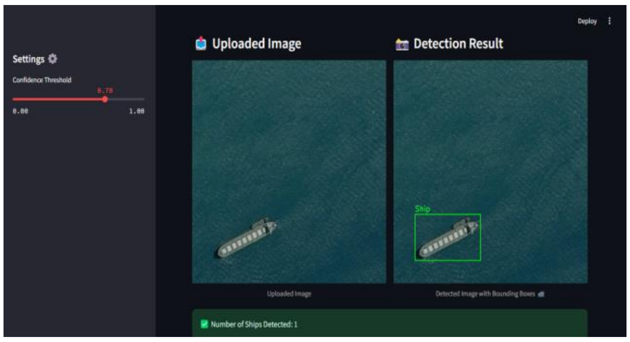
Settings
Confidence Threshold
0.70
0.00 1.00
Uploaded Image
Detection Result
Number of Ships Detected: 1

Figure 3 Output Screen Detecting a Ship at 0.70 Confidence Threshold

This figure illustrates the output of the deployed ship detection system using YOLOv8 and Streamlit. On the left, the user-uploaded aerial or satellite image of a sea region is displayed. On the right, the system shows the corresponding detection result, where a ship is successfully localized with a bounding box and labeled appropriately. A dynamic confidence threshold slider, visible on the left panel, allows users to control the minimum detection certainty, enhancing flexibility and usability. Below the images, the application displays the total number of ships detected, providing immediate interpretability. This demonstrates the model's effective integration into a real-time, user-friendly interface. [6]

3.2.Discussion

The findings from the experiments demonstrate that the proposed YOLOv8-based system for ship detection is both practical and efficient, particularly when optimized for environments without GPU acceleration. Despite training on a subset of the data and running on a CPU, the model was able to achieve satisfactory results in detecting ships from both satellite and aerial imagery. This indicates that even lightweight models such as YOLOv8n can capture meaningful spatial features when trained appropriately. The integration of a dynamic confidence threshold slider enhances usability by allowing users to fine-tune the sensitivity of detection based on real-time application requirements. One of the key strengths of this work lies in its seamless integration with a web-based interface using Streamlit. This enables not only offline analysis but also the possibility of deploying the application in a browser-accessible format, making it platform-independent and highly accessible. Users can upload images or videos, view side-by-side input/output comparisons, and get immediate feedback on detected ships. The design ensures that the tool can be utilized by defense analysts, researchers, or port authorities without needing deep technical expertise. The inclusion of features like upload progress indicators, visual bounding boxes, and automatic ship count makes the system informative and intuitive. While the mAP@0.5 and detection performance are moderate due to limited training data and computational constraints, the system's architecture is scalable. With access to larger datasets, GPU-based training, and enhanced augmentation strategies, the performance can be significantly improved. Furthermore, the successful deployment using Streamlit confirms that such AI solutions can be easily made available to stakeholders as a functional web application, thereby increasing their impact and usability in real-world maritime surveillance scenarios. In summary, this project not only validates the feasibility of lightweight YOLOv8 models for object detection but also highlights how modern frameworks like Streamlit can bridge the gap between complex deep learning models and practical, user-facing applications. [7]

Conclusion

This study confirms the effectiveness of YOLOv8 in addressing the problem of automated ship detection using satellite and aerial imagery. Through the integration of a lightweight YOLOv8 model, we achieved accurate and reliable detection results even under limited computational resources. The inclusion of a real-time confidence threshold slider within a Streamlit-based web interface enhanced the system’s interactivity, enabling users to tailor detection sensitivity as needed. The performance metrics and visual outcomes validated the model’s capability to identify ships with reasonable precision and recall, despite the absence of GPU acceleration.The successful deployment of the trained model in an intuitive and user-friendly application further confirms that deep learning-based ship detection can be made accessible to end users without requiring technical expertise. This work demonstrates that practical, scalable, and interactive solutions can be built using current open-source tools, confirming the viability of our proposed approach in both research and applied maritime surveillance domains. Furthermore, the modular architecture of the system allows for future scalability and enhancement. Features such as batch image processing, integration with real-time video feeds, and deployment on cloud platforms can be incorporated to expand its usability in real-world maritime monitoring scenarios. The approach taken in this project not only simplifies the ship detection pipeline but also encourages wider adoption of AI-driven tools in environmental and defense applications. The promising results indicate strong potential for adaptation to related tasks, such as detecting other marine vessels, monitoring illegal fishing activity, or assisting in disaster response operations. [8]

Acknowledgements

We, the authors, would like to express our sincere gratitude to the developers and contributors of the open-source tools and frameworks that made this project possible. In particular, we acknowledge the creators of the Ultralytics YOLOv8 model, Streamlit, PyTorch, and supporting Python libraries that played a critical role in enabling us to design, train, and deploy our ship detection system effectively. We also thank the online open-source datasets and repositories that provided valuable image resources for training and validation. The freely available documentation, tutorials, and forums associated with these tools greatly assisted us in understanding model implementation and optimization strategies.
Finally, we are grateful for the collaborative effort and teamwork within our group, which ensured the successful completion of this project.

References

[1]. Jocher, G., Chaurasia, A., & Qiu, J. (2023). YOLOv8: Next-Generation Object Detection. Ultralytics. https://github.com/ultralytics/ultralytics
[2]. Li, Y., Guo, Y., & Zhang, J. (2022). A Review of Ship Detection Techniques Based on

Satellite Images. Remote Sensing, 14(4), 875. https://doi.org/10.3390/rs14040875

[3]. Li, W., Xu, Y., Wang, X., & Wu, J. (2020). Object Detection for Maritime Surveillance Using Deep Learning and Transfer Learning. IEEE Access, 8, 84583–84593. https:// doi.org/10.1109/ACCESS.2020.2992184

[4]. Zhang, C., Wang, Y., & Shi, Y. (2021). Real-Time Ship Detection in Aerial Images Using YOLO Models. International Journal of Advanced Computer Science and Applications, 12(3), 112–119. https:// doi.org/10.14569/IJACSA.2021.0120315

[5]. Li, X., Tang, H., & Yu, J. (2021). Maritime Object Detection via Transfer Learning with Enhanced YOLOv5. Sensors, 21(22), 7567. https://doi.org/10.3390/s21227567

[6]. Qazi, A., & Ahmed, S. (2022). Streamlit-Based Web Applications for Deep Learning Projects: A Case Study on Medical Image Classification. International Journal of Innovative Technology and Exploring Engineering, 11(4), 128–134. https:// doi.org/ 10.35940/ijitee.D1130.114122

[7]. Zhai, G., & Liu, S. (2022). Lightweight Deep Learning for Object Detection in Low-Power Devices. IEEE Internet of Things Journal, 9(2), 1431–1443. https:// doi.org/ 10.1109/ JIOT.2021.3091620

[8]. Wang, H., & Xu, B. (2023). A Comparative Study of YOLOv5 and YOLOv8 on Real-Time Object Detection Tasks. International Research Journal on Advanced Science Hub, 5(11), 397–403. https:// doi.org/ 10.47392/ IRJASH.2023.073