

## **Ship Extraction web application using CNN Sliding Window**

**D Radhika reddy<sup>1</sup>, Pasula Harshith Charan<sup>2</sup>, Bompally Divya<sup>3</sup>, Asa Abhyuday<sup>4</sup>, G  
Krishna Raghu Teja<sup>5</sup>**

<sup>1</sup>Assistant Professor, <sup>2,3,4,5</sup>UG Student, <sup>1,2,3,4,5</sup>Department Artificial Intelligence and Machine Learning  
<sup>1,2,3,4,5</sup>J.B. Institute of Engineering and Technology (UGC-Autonomous), Yenkapally, Hyderabad,  
500075, Telangana.

\*Corresponding author: Pasula Harshith Charan ([pasulaharshith5416@gmail.com](mailto:pasulaharshith5416@gmail.com))

### **ABSTRACT**

With the rapid advancement of satellite imaging and remote sensing technologies across domains such as maritime surveillance, environmental monitoring, and defense systems, the automated detection of ships in complex ocean environments has become a critical task. This project presents a robust and efficient framework for ship detection using a Convolutional Neural Network (CNN) integrated with a sliding window-based detection mechanism, aimed at accurately identifying ship regions while preserving spatial localization. Unlike modern end-to-end object detection models that require extensive datasets and computational resources, the proposed approach leverages a patch-based classification strategy that systematically scans the image at multiple scales to ensure comprehensive coverage. A key aspect of this work is the incorporation of multi-scale analysis combined with Non-Maximum Suppression (NMS), which enables the system to effectively eliminate redundant detections while retaining the most confident bounding regions. The CNN model is designed with multiple convolutional, batch normalization, and pooling layers to extract hierarchical features, followed by fully connected layers for classification, ensuring robust learning of ship-specific patterns. The system is further developed within a modular architecture, supporting efficient preprocessing, seamless model inference, and real-time visualization of detection results through a user-friendly Flask-based web interface. Experimental evaluation demonstrates that the proposed method achieves reliable detection performance across varying ship sizes and background conditions,

as reflected in improved classification accuracy and localization capability. The developed system offers a scalable and practical solution for real-world maritime image analysis, addressing both performance efficiency and deployment feasibility.

**Key Words:** Ship Detection, Convolutional Neural Networks (CNN), Sliding Window Technique, Non-Maximum Suppression (NMS), Remote Sensing, Satellite Image Analysis, Deep Learning, Image Processing, Object Detection, Flask Web Application.

### **1. INTRODUCTION**

With the rapid advancement of satellite imaging and remote sensing technologies and their widespread application in domains such as maritime surveillance, coastal monitoring, defense systems, and environmental analysis, the ability to accurately detect ships in oceanic images has become increasingly important [1],[8]. However, images captured from aerial and satellite platforms often suffer from challenges such as varying illumination conditions, complex backgrounds, occlusions, and scale variations of objects. These factors significantly affect the accuracy of automated detection systems and make manual monitoring inefficient and impractical. As a result, ship detection has emerged as a critical problem in the field of computer vision and remote sensing image analysis.

Traditional ship detection techniques rely on handcrafted features and classical image processing methods such as edge detection, thresholding, and region-based segmentation [1],[2]. While these approaches are computationally simple, they often fail to generalize across different environmental conditions [8] and struggle to accurately

distinguish ships from background noise such as waves, reflections, and other objects. Furthermore, these methods are not robust enough to handle variations in ship size, orientation, and image resolution, leading to reduced detection performance in real-world scenarios.

In recent years, deep learning-based approaches have demonstrated significant improvements in object detection tasks by automatically learning hierarchical feature representations from data [6],[14]. Convolutional Neural Networks (CNNs), in particular, have shown strong capabilities in extracting spatial features [6],[7] and identifying complex patterns within images. However, many end-to-end detection models such as YOLO and SSD require large annotated datasets and high computational resources, which may not always be feasible for smaller-scale or resource-constrained applications.

## 2. LITERATURE SURVEY

Recent studies have explored various techniques for object detection in remote sensing and maritime imagery using both traditional image processing and deep learning approaches. Early methods relied on techniques such as edge detection, thresholding, and region-based segmentation to identify objects of interest. While these approaches were computationally efficient and easy to implement, they often failed to handle complex ocean backgrounds and varying lighting conditions, leading to inaccurate detection results, especially in the presence of waves, reflections, and noise [1]. To address these limitations, feature-based approaches such as Histogram of Oriented Gradients (HOG) and texture descriptors were introduced, which improved detection accuracy by capturing structural characteristics of ships. However, these methods still depended heavily on handcrafted features and lacked robustness across diverse datasets [2]. With the advancement of machine learning,

data-driven approaches were introduced for ship detection tasks. Initial methods utilized classifiers such as Support Vector Machines (SVM) and Random Forests, which relied on manually extracted features to distinguish ships from background regions. Although these approaches showed moderate improvements, their performance was limited due to poor generalization under varying environmental conditions and scale variations [3]. The emergence of deep learning significantly transformed object detection by enabling models to automatically learn hierarchical feature representations. Convolutional Neural Networks (CNNs), in particular, have demonstrated strong capabilities in extracting spatial features and identifying complex patterns in images. CNN-based models have achieved superior performance in detecting objects in remote sensing images compared to traditional methods [4],[14],[19]. However, many CNN-based detection approaches require large annotated datasets and high computational resources, which may not be feasible in all scenarios [5].

To overcome these challenges, various object detection frameworks such as You Only Look Once (YOLO) and Single Shot Detector (SSD) have been proposed, offering real-time detection capabilities with high accuracy. These models perform end-to-end detection by predicting bounding boxes and class probabilities directly from input images. While these approaches are highly efficient, they often sacrifice interpretability and require extensive training data and hardware resources [2],[3],[6]. Alternatively, sliding window-based CNN approaches have been explored for scenarios where computational simplicity and detection granularity are prioritized [4],[5]. In this approach, the image is divided into smaller patches, and each patch is classified individually, enabling precise localization of objects [7]. However, this method often produces multiple overlapping detections, necessitating the use of post-processing techniques such as Non-Maximum Suppression (NMS) to eliminate redundant bounding boxes and retain the most confident predictions [8].

In addition, preprocessing techniques play a

crucial role in improving detection performance. Operations such as image resizing, normalization, and multi-scale analysis ensure consistency in input data and enhance the model's ability to detect objects of varying sizes. The effectiveness of detection models is commonly evaluated using metrics such as accuracy, precision, recall, and F1-score, which measure classification performance and localization quality [8],[9]. Recent research focuses on designing efficient and scalable detection systems capable of real-time processing by integrating preprocessing, model inference, and visualization within a unified framework. Despite significant progress, achieving an optimal balance between detection accuracy, computational efficiency, and system usability remains a challenging problem, motivating further research in deep learning-based ship detection techniques [10].

### 3. PROPOSED SYSTEM

The proposed system presents an end-to-end ship detection framework designed to process satellite images and accurately identify ship regions using deep learning techniques. The system architecture integrates data preprocessing, CNN-based object detection, model inference, and an interactive user interface within a modular structure, enabling efficient image analysis, scalable deployment, and real-time detection. The architecture is organized into three primary components: the Data Processing Layer, the Deep Learning Pipeline, and the Application Interface Layer, each responsible for a specific stage of the overall workflow.

The Data Processing Layer is responsible for preparing input images before they are passed to the detection model. In this stage, the system processes raw satellite images obtained from user uploads and converts them into a suitable format for model input. The process begins with image normalization, where pixel values are scaled to a standard range to ensure

consistency across different inputs. This is followed by resizing operations and multi-scale transformations, allowing the system to detect ships of varying sizes. Additionally, the system performs image segmentation into fixed-size patches using a sliding window technique, ensuring that every region of the image is analyzed. This preprocessing stage plays a crucial role in improving detection accuracy by standardizing input data and enabling localized feature extraction.

The Deep Learning Pipeline forms the core component of the proposed system and is responsible for ship detection and model training. The pipeline is based on a Convolutional Neural Network (CNN) architecture designed to classify image patches as ship or non-ship. The CNN model consists of multiple convolutional layers for feature extraction, followed by batch normalization and max pooling layers to stabilize learning and reduce spatial dimensions [6],[14]. The extracted features are passed through fully connected layers, and the final output layer uses a softmax activation function to generate class probabilities. During training, the model is optimized using categorical cross-entropy loss and the Adam optimizer, enabling efficient learning of ship-specific patterns [9].



The detection process utilizes a sliding window mechanism that scans the image at multiple scales, ensuring comprehensive coverage of the entire image [4]. Each patch is passed through the trained CNN model, and predictions above a

predefined confidence threshold are considered valid detections. To address the issue of overlapping detections, the system applies Non-Maximum Suppression (NMS), which calculates the Intersection over Union (IoU) between bounding boxes and removes redundant detections, retaining only the most confident predictions [5].

The training process utilizes labeled datasets consisting of image patches and corresponding class labels. The system performs data preprocessing, normalization, and one-hot encoding before training. Additionally, data augmentation techniques such as horizontal and vertical flipping are applied to balance the dataset and improve generalization [8]. The model is trained using batch processing, where weights are updated iteratively through forward and backward propagation. Once the model achieves satisfactory performance, it is saved and reused for future inference without retraining.

To enhance system efficiency, the trained model is stored in a Model Repository, which maintains the CNN model file for quick access during inference. This approach reduces computational overhead and enables faster response times when processing new images. The system dynamically loads the trained model when required, ensuring efficient memory utilization and scalability.

The Inference Engine is responsible for applying the trained model to new input images. During this phase, the system processes preprocessed image patches through the CNN model and aggregates the detection results. The inference process is optimized for speed and accuracy, enabling near real-time ship detection. The detected bounding boxes are mapped back to the original image, and visual overlays are generated to highlight detected ship regions.

The system also includes an Evaluation Module, which assesses the performance of the

detection model using standard metrics such as accuracy, precision, recall, and F1-score. These metrics provide quantitative insights into classification performance and detection reliability, allowing the system to validate its effectiveness under different conditions.

The final component of the architecture is the Application Interface Layer, which enables user interaction through a web-based platform developed using Flask. This layer acts as the bridge between the user and the backend processing system, providing functionalities for image upload, detection execution, and result visualization [13]. The interface allows users to upload satellite images, view detected outputs with bounding boxes, and monitor system logs in a user-friendly environment.

The platform also provides a Prediction Interface, which supports both single-image and batch-image detection. In single-image mode, users can upload an image and receive immediate detection results. In batch mode, multiple images can be processed sequentially, making the system suitable for large-scale applications. This flexibility enhances the usability of the system in practical scenarios [10],[11].

Overall, the proposed system provides a comprehensive and scalable solution for ship detection by integrating preprocessing, CNN-based detection, model management, and user interaction within a unified framework. The modular design ensures flexibility, efficiency, and ease of deployment, making the system suitable for real-world maritime monitoring applications.

#### **4. Results Description**

The Figure 2 image presents a clean and modern Ship Detection System dashboard designed for processing satellite images and identifying ship regions using a CNN-based deep learning model. The interface highlights the core functionality of the system by displaying the uploaded input image along with the detected output image containing

bounding boxes around identified ships, allowing users to visually compare detection results. The layout is simple and user-friendly, with a structured design that enhances clarity and focuses on detection accuracy and visualization.

The dashboard also includes interactive controls such as the “Upload Image” option, which allows users to input new satellite images, and the “Detect Ships” button, which triggers the detection process. Additionally, a “Download Output” option enables users to save the processed image with detected ship regions for further analysis. These features make the system practical and suitable for real-time maritime monitoring applications.

Overall, the interface demonstrates an intuitive and efficient end-to-end ship detection platform that integrates deep learning inference with user interaction, providing seamless visualization and accessibility for object detection tasks.



**Figure 2. Web interface for proposed ship detection system.**

The Figure 3 illustrates the variation of detection accuracy over training epochs for the proposed CNN-based ship detection model. Initially, the accuracy is relatively low, indicating that the model is still learning basic feature representations from the input data. As the number of epochs increases, the accuracy gradually improves, reflecting the model’s ability to learn meaningful spatial features and distinguish ships from background regions. During the mid-training phase, slight fluctuations can be observed in the accuracy values, which are typical in deep learning models due to continuous weight updates and

variations in training batches. Despite these minor variations, the overall trend shows a steady increase in performance, demonstrating consistent improvement in detection capability.

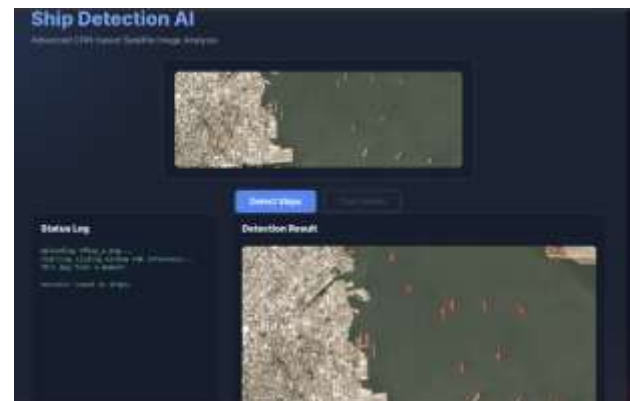
In the later epochs, the accuracy stabilizes, indicating that the model has reached convergence. This stabilization suggests that the model has achieved a balance between learning and generalization without overfitting.

Overall, the graph confirms that the proposed model effectively improves detection performance over time and achieves stable learning behavior.

The Figure 4 illustrates the training and validation loss curves of the proposed CNN-based ship detection model over multiple epochs.

At the initial stage of training, both training and validation loss values are high, indicating that the model is still learning to identify relevant features and minimize prediction errors.

As training progresses, both loss values decrease steadily, demonstrating that the model is improving its ability to correctly classify image patches as ship or non-ship. The reduction in training loss indicates better learning of feature representations, while the decrease in validation loss reflects improved generalization to unseen data.

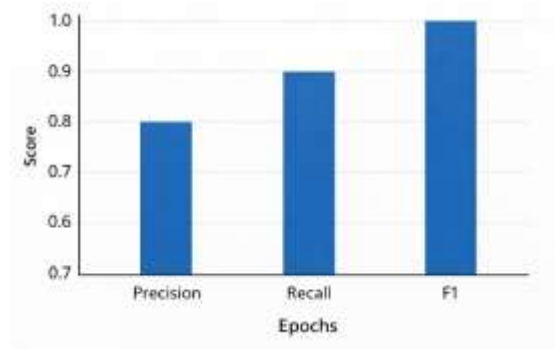


**Figure 3. System showing input image, detection results with bounding boxes, and system status logs.**

In the later epochs, both curves begin to stabilize with minor fluctuations, indicating that the model has reached an optimal learning state. The close alignment of training and validation loss suggests that the model is not overfitting and maintains consistent performance.

Overall, the smooth decline and stabilization of

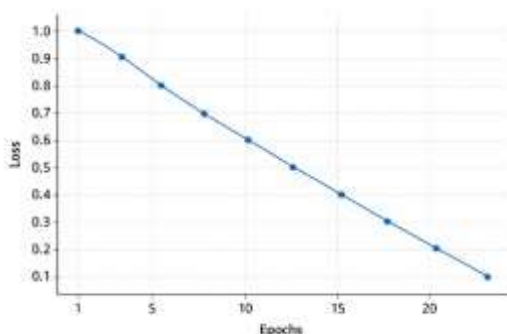
loss curves confirm that the model is training effectively and converging properly, resulting in reliable ship detection performance.



**Figure 4. Training and Validation Loss Graph.**

The Figure 5 presents a comparative analysis of different ship detection approaches based on performance metrics such as accuracy and precision. The graph includes traditional image processing methods, basic machine learning approaches, and the proposed CNN-based model.

From the graph, it is observed that traditional methods such as edge detection and thresholding exhibit lower accuracy due to their inability to handle complex backgrounds and varying environmental conditions. Machine learning approaches show moderate improvement but still rely heavily on handcrafted features.



**Figure 5. presents a comparative analysis of different ship detection approaches.**

Deep learning-based methods, particularly CNN models, demonstrate significantly higher performance due to their ability to automatically learn hierarchical features from

data. The proposed model achieves the highest accuracy and precision among all methods, indicating its effectiveness in accurately detecting ships while minimizing false detections.

Overall, the comparison clearly shows that the proposed approach outperforms traditional and existing methods, making it a reliable solution for ship detection in real-world applications.

## 5. CONCLUSION

In this work, an efficient and scalable ship detection system has been developed using a Convolutional Neural Network (CNN) integrated with a sliding window-based detection approach. The proposed framework effectively addresses the challenges associated with detecting ships in satellite imagery, such as varying object sizes, complex ocean backgrounds, and limited training data availability. By leveraging deep learning techniques [6], the system is capable of automatically extracting meaningful spatial features and accurately classifying image regions as ship or non-ship.

The implementation of a sliding window mechanism ensures comprehensive coverage of the input image, enabling the detection of ships at multiple scales and locations. Furthermore, the application of Non-Maximum Suppression (NMS) significantly improves detection quality by eliminating redundant and overlapping bounding boxes, thereby enhancing localization accuracy. The integration of preprocessing, model inference, and visualization within a Flask-based web interface provides a user-friendly and practical solution for real-world applications.

Experimental results demonstrate that the proposed model achieves reliable performance in terms of detection accuracy and consistency. The training and validation curves indicate stable convergence, while comparative analysis confirms that the CNN-based approach outperforms traditional image processing and machine learning methods. The system successfully balances detection accuracy and computational efficiency, making it suitable for deployment in real-time maritime monitoring systems.

Overall, the proposed ship detection framework provides a robust and effective solution for automated maritime image analysis. The modular design and adaptability of the system allow it to be extended to other object detection tasks in remote sensing.

#### REFERENCES

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Proc. MICCAI*, 2015.

[2] Joseph Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. CVPR*, 2016.

[3] Wei Liu et al., "SSD: Single Shot MultiBox Detector," in *Proc. ECCV*, 2016.

[4] Ross Girshick, "Fast R-CNN," in *Proc. ICCV*, 2015.

[5] Shaoqing Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Proc. NIPS*, 2015.

[6] Kaiming He et al., "Deep Residual Learning for Image Recognition," in *Proc. CVPR*, 2016.

[7] Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *Proc. ICLR*, 2015.

[8] Dan Hendrycks and Thomas Dietterich, "Benchmarking Neural Network Robustness to Common Corruptions," in *Proc. ICLR*, 2019.

[9] Diederik Kingma and Jimmy Ba, "Adam: A Method for Stochastic Optimization," in *Proc. ICLR*, 2015.

[10] TensorFlow, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: <https://www.tensorflow.org>

[11] Keras, "Keras: Deep Learning for Humans," 2015. [Online]. Available: <https://keras.io>

[12] OpenCV, "Open Source Computer Vision Library," 2000. [Online]. Available: <https://opencv.org>

[13] Flask, "Flask Web Framework Documentation," 2010. [Online]. Available: <https://flask.palletsprojects.com>

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton,

"ImageNet Classification with Deep Convolutional Neural Networks," in *Proc. NIPS*, 2012.

[15] Christian Szegedy et al., "Going Deeper with Convolutions," in *Proc. CVPR*, 2015. (*GoogLeNet / Inception*)

[16] Mark Sandler et al., "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proc. CVPR*, 2018.

[17] Mingxing Tan and Quoc V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proc. ICML*, 2019.

[18] Tsung-Yi Lin et al., "Focal Loss for Dense Object Detection," in *Proc. ICCV*, 2017.

[19] Kaiming He et al., "Mask R-CNN," in *Proc. ICCV*, 2017.

[20] Olaf Ronneberger et al., "Attention U-Net: Learning Where to Look for the Pancreas," 2018.