Drone Simulation Approach for Maritime Search and Rescue

Spandan Kiran Vaidya [a], Sakthivel Velusamy [a*], Parit Gupta [a], Rupankar Majumdar [a], Jeevan S[b]

[a] *School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, Tamilnadu, India*

[b,] *School of Mechanical Engineering, Vellore Institute of Technology, Chennai, Tamilnadu, India*

**Corresponding author, E-mail: sakthivel.v@vit.ac.in, Phone: +91 9843677515*

Abstract

Maritime search and rescue (SAR) operations are critical, time-sensitive, and often require rapid and large-scale area scanning, posing a significant challenge for manual & semi-automated methods that struggle to meet these needs. The study focuses on developing an integrated advanced machine vision system to enhance SAR operations & effectiveness, leveraging drone technology to cover large areas and execute missions with greater precision. The objective of this research is to explore the applications of drone technology in SAR operations and evaluate advanced object detection models using simulated scenarios. Utilizing Unreal Engine 4.2, a custom Maritime SAR scenario was simulated to test and compare the performance of advanced object detection models, specifically the YOLOv8 and YOLOv11 series trained on the SeaDronesSee dataset to support the detection and tracking tasks that are significant for SAR operations. The evaluation revealed that the YOLOv11m, YOLOv8l, and YOLOv8x models exhibited the best performance, with mAP50 scores of 0.692, 0.687, and 0.680 respectively, with the YOLOv11 outperforming previous models in terms of accuracy, and utilizing lesser GFLOPs, making it resource efficient & an ideal choice for deployment on edge devices for drones. This conclusion drawn from the results emphasizes on the highly promising potential of the YOLOv11 series, integrating advanced machine vision for maritime search & rescue operations, providing a scalable and accurate solution for real-time detection in dynamic environments. The findings indicate that integrating advanced machine vision with drone technology can significantly enhance maritime SAR operations.

Keywords: Maritime Search, Drones, YOLOv8, YOLOv11, Unreal Engine, Machine Vision, Rescue Operations.

1. Introduction

The open sea, vast and unpredictable, has long posed a challenge to humanity, with its boundless horizons and uncharted depths, it often becomes the stage for perilous accidents, from shipwrecks to individuals struggling against waves. Maritime Search and Rescue (SAR) operations, though critical in saving lives, are frequently hampered by the vast scale of the seas and the difficulty of finding the exact locations of the victims in time-sensitive scenarios. Traditional methods of search and rescue, be it manual or semi-automatic, struggle to meet the demands of these urgent situations, highlighting the need for innovative solutions.

This is where modern technology comes in, offering hope and endless possibilities. Among the most promising advancements in modern technology is the use of unmanned aerial vehicles (UAVs), commonly known as drones. A machine that can fly independently without carrying any human pilot and can be pre-programmed or remotely controlled is identified as an unmanned aerial vehicle or UAV. Drones, which belong to the umbrella term UAV, are flying robots that can be classified based on their structural design, functional capability, and use case [1]. Drones can be classified based on their size as mini, small, and large [2]. Large drones are confined to military applications while the small and mini drones, weighing up to 25 kg, can be used for several commercial applications like photography, videography, surveys, architectural scans, pesticide spraying, etc. Research in small and mini-sized drones has increased significantly over the years [3].

Small drones have been shown to enhance the functionality and efficiency of a service. Some sectors that have benefitted from the use of drones are healthcare [4], photography and videography [5], agriculture [6], road damage detection [7], etc. A drone that can fly independently of constant human control by following the commands of its onboard autopilot software is said to be autonomous. Inputs like GPS coordinates, inertial measurement units, or IMU data like acceleration, are provided to the control loop that processes to produce actuator outputs responsible for flying a drone desirably [8].

Maritime search and rescue (SAR) operations are time-critical and often, deal with regions where the approach is difficult [9]. The vastness of the sea may pose a great challenge for SAR boats to identify the drowning victims. Current manual or semi-automated methods lack efficiency in timely locating drowning victims or rescue boats. This creates a need for a system that can quickly and accurately identify the position of swimmers and rescue boats in real time. This is where the capabilities of a drone can be extended to maritime search and rescue operations. With the sea becoming an important mode of transportation for trade, the risk of maritime hazards also increases. Realizing the potential of drones in mapping vast areas with much higher efficiency and accuracy than the traditional SAR approaches, a rapid response to a victim can impact its survival rate substantially [10].

Computer simulations play a major role in designing rescue operations. A computer simulation for a maritime SAR operation incorporating drones can help technicians realize the strengths and weaknesses of the current state of technology at dispense. Simulations can lead to iterative developments in mission planning, resulting in a significant reduction of cost, time, and manual effort. Simulations can also help benchmark performance and presume practical expectations from a real-world SAR.

Unreal Engine (UE) [23] is a game engine developed by Epic Games that proves to be a powerful tool for developing drone simulations. This research utilizes UE version 4.2 to develop the test-SAR scenario and to execute the simulation. An object detection model developed using Ultralytics' YOLO framework powers the 'Search' in the SAR.

This study tries to find novelty in creating a virtual environment to simulate real-world scenarios for validating and testing pre-trained machine learning algorithms. This breakthrough can help reduce time and optimize dataset creation and development costs for multiple test scenarios.

1.1 Background Motivation

For many decades, the Mediterranean Sea has been witnessing boats and vessels overcrowded with seafarers [11], posing a great threat to human lives. There have been several tragic incidences in the Mediterranean Sea leading to deaths of humans due to drowning. On 1 September 2012, the sea saw a death toll of 136. This number increased to 339 in October 2023, perhaps more recently. A similar event happened in late 2023, where the Maltese Navy rescued 150 victims of drowning, but failed to save the lives of 34 others [12]. Realizing the speed and efficiency of a UAV-based search and rescue in situations of distress has motivated the authors of this paper to develop a simulation-based approach and an object detection model for a maritime SAR.

1.2 Dataset

The dataset used in the research is an adoption of the SeaDronesSee [20] dataset generated by the researchers at the University of Tübingen which has been used to train the object detection model. The dataset contains 10474 images containing 5 classes namely, 'boat', 'buoy', 'jetski', 'life_saving_appliances', and, 'swimmer.' Recorded by 20 test subjects, this dataset contains footage of open waters in different lighting conditions [13]. The dataset has been augmented with 'cutout augmentation.' Cutout is a technique used in image augmentation that randomly masks out square regions of input images [14]. A split of 70-20-10 (in percentages) has been carried out on the entire dataset to generate the training, validation, and test subsets resulting in a split of 7322, 2077, and 1075 images respectively which has been represented in Table 1.

Table 1. Dataset Partitioning

Sub-set	**Proportion**	**Images**
Train	70%	7,322
Val	20%	2,077
Test	10%	1,075

1.3 YOLO Object Detection

YOLO, which stands for, You Look Only Once is a widely used algorithm, popular for its object detection capabilities. Since the inception of the YOLO series in 2015 [15], there have been successive versions of YOLO improving in speed and accuracy [16]. The object detection model employed in this research has been trained on Nano, Small, Medium, Large, and, Extra Large models of the YOLOv8 and YOLOv11 framework. This research has involved multiple iterations of model building with parameter and structure adjustments to get a suitable model with ground-breaking evaluation metrics.

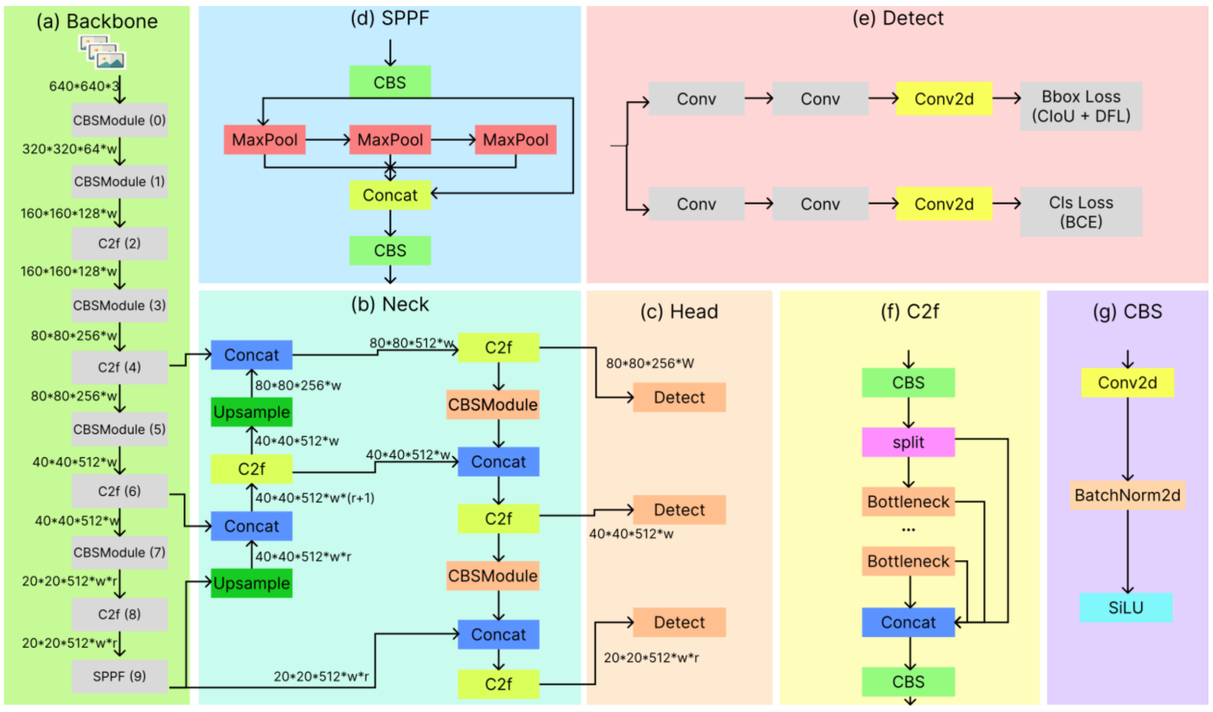


Figure 1. Schematic figure of the network structure of YOLOV8

YOLOv8 employs a modified CSPDarknet53 architecture [27] as its backbone network, where input features are progressively down-sampled five times to produce feature maps at five distinct scales, referred to as B1 through B5 (illustrated in Figure 1a). Instead of the original Cross Stage Partial (CSP) module, YOLOv8 integrates the C2f module, depicted in Figure 1f. This C2f module incorporates a gradient shunt connection, which enhances the flow of information within the feature extraction process while keeping the architecture lightweight.

The CBS module, as depicted in Figure 1g, is responsible for performing convolution operations, followed by batch normalization, and finally activates the information stream using the Sigmoid Linear Unit (SiLU) activation function to produce the output. At the end of the backbone, a Spatial Pyramid Pooling Fast (SPPF) module is employed to adaptively pool the feature maps into a fixed-size representation. Compared to traditional Spatial Pyramid Pooling (SPP), the SPPF module reduces computation and latency by linking three max-pooling layers sequentially, as demonstrated in Figure 1d.

Inspired by PANet [28], YOLOv8 incorporates a PAN-FPN structure at its neck, as depicted in Figure 1b. Unlike YOLOv5 and YOLOv7, YOLOv8 removes the convolution operation following the up-sampling step in the PAN structure, preserving performance while minimizing model complexity.

The PAN-FPN design merges features at two different scales: P4-P5 from the PAN structure and N4-N5 from the FPN structure. The conventional FPN relies on a top-down approach to blend deep semantic information. While this approach enriches semantic data by fusing B4-P4 and B3-P3, it sometimes sacrifices precise object localization. To address this, PAN-FPN integrates PAN to enhance positional learning through the fusion of P4-N4 and P5-N5. This combination creates a dual-path network (top-down and bottom-up), enabling feature fusion that balances positional details and semantic depth, resulting in richer and more complete feature representations.

In YOLOv8, the detection head has a decoupled design, as depicted in Figure 1e. This decoupled structure consists of separate branches for classification and bounding box regression, each utilizing distinct loss functions. For classification, ‘Binary Cross Entropy’ (BCE) is used. A combination of ‘Distribution Focal Loss’ (DFL) and ‘Complete Intersection over Union’ (CIoU) loss is applied for bounding box regression.

YOLOv8 uses an anchor-free approach, simplifying the distinction between positive and negative samples. Additionally, it uses the ‘Task-Aligned’ Assigner to dynamically allocate samples. This enhances both the detection accuracy and the robustness of the model.

YOLO11 builds upon the foundation of YOLOv8, offering enhanced detection and segmentation capabilities with greater efficiency and accuracy on benchmark datasets. It incorporates an advanced backbone and neck structure to improve feature extraction, enabling more precise object detection. The architecture and training methodologies have been optimized for faster processing while maintaining efficiency.

Despite its reduced complexity, YOLOv11 achieves higher mean Average Precision (mAP) scores on the COCO dataset with fewer parameters compared to YOLOv8m. Additionally, its design ensures broad adaptability, making it compatible with a variety of platforms, including edge devices and cloud systems. Supporting tasks ranging from basic object detection to complex oriented detection and segmentation, YOLOv11 provides flexible deployment options for inference and model export. [33]

2. Literature Survey

2.1 Overview of Object Detection in Rescue Operations

Utilization of drones for the detection of victims of falls and injuries due to adventurous activities like skiing, hiking, mountain biking, etc., is discussed in [30] with an image-based, YOLOv4 object detection approach. Automatic human detection in SAR scenarios by utilizing a Thermal Infrared (TIR) camera to combat the limitations of a vision-based detection system is discussed in [31]. Convolutional neural network (CNN) models designed for detecting ground objects from aerial views of disaster aftermaths in detailed in [32]. The described models (in [32]) can identify essential ground features such as building roofs (both intact and damaged), vehicles, vegetation, debris, and flooded areas, which were trained on a custom aerial video dataset, Volan2018.

2.2 Summary of SeaDronesSee dataset in Rescue tasks

The SeaDronesSee dataset [20] has been developed out of the need for a suitable dataset for maritime SAR missions. The earlier datasets that have captured the maritime environment, targeted synthetic aperture radar imagery for remote sensing applications [17]. These datasets have relied on satellite-based image generation which provides only the top-view, naturally. These datasets thus become more useful for identifying large-sized ships and not for smaller objects like swimmers. Moreover, satellite-based imagery poses its own set of limitations to cloudy weather [13]. In developing the SeaDronesSee dataset, images, and videos of the dedicated classes (boat, jet ski, buoy, life-saving appliances, and, swimmer) were captured with an RGB footage of 3840 * 2160 px to 5456 * 3632 px resolution [13].

2.3 Summary of Object Detection models utilizing the SeaDronesSee dataset

There are several object detection models utilizing the SeaDronesSee dataset. Some models developed on the variants of Faster R-CNN, Cascade R-CNN, and RetinaNet have resulted in AP50 scores ranging from 0.301 to 0.718, the latter is the result of FR.ResNeXt-101-FPN-heuristic model [25]. Modifications to the default YOLOv8 models have resulted in AP50 scores ranging from 0.379 to 0.591 [26]. The complexity of the dataset, due to the presence of very small objects like those belonging to the 'swimmer,' and 'life_saving_applainces' classes, makes object detection models perform poorly against the validation dataset.

3. Methodology

3.1 Dataset Preparation

An iteration of the SeaDronesSee dataset [20], originally containing 5,630 images with annotations is selected. These images have been recorded by 5 camera systems as listed in Table 2. This effort has been made to nullify the effect of camera biases. The use of 3 drones have been made to capture the data; DJI Matrice 100, DJI Matrice 210, DJI Mavic 2 Pro, and a fixed-wing Trinity F90+, developed by Quantum Systems [13].

Table 2. Specifications of cameras used in generating the SeaDronesSee dataset

Camera	Resolution	Purpose
Hasselblad L1D-20c	3,840 * 2,160	Video capture at 30 fps
MicaSense RedEdge-MX	1,280 * 960	Multi-spectral capture at 1 fps
Sony UMC-R10C	5,456 * 3,632	Image capture
Zenmuse X5	3,840 * 2,160	Video capture at 30 fps
Zenmuse XT2	3,840 * 2,160	Video capture at 30 fps

The annotations have been made by DarkLabel [18], a free labeling tool. The original annotations were created into the following classes; swimmer (person in water without life jacket), floater (person in water with life jacket), swimmer† (person on boat without life jacket), floater† (person on boat with life jacket), and boats.
The dataset used in this research is the Roboflow SeaDronesSee v10 [19], adopted from the original SeaDronesSee dataset [20]. The Roboflow dataset has some key modifications to the original dataset.

Cutout augmentation and auto orientation have enabled the expansion of the dataset to 10,474 images. The original classes have been re-distributed to generate 5 classes namely, namely, 'boat', 'buoy', 'jetski', 'life_saving_appliances', and, 'swimmer.' The 'swimmer' classes have been generated by merging the 4 classes namely, swimmer, floater, swimmer †, and floater† from the original SeaDronesSee dataset [20], whose distribution of occurrences has been identified in Table 3, along with the distribution of instances of each class across in the training dataset which is depicted in Figure 2.

Table 3. Frequency of class image across dataset images

Class	**Images**
boat	6,782
buoy	4,073
jetski	2,648
life_saving_appliances	856
swimmer	8,185

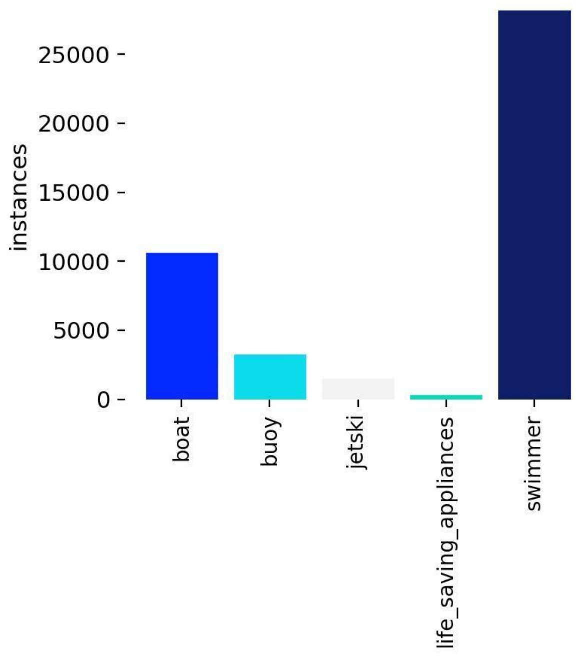


Figure 2. Graphical representation of the instances of various classes in the training dataset

3.2 Model Configurations

The configurations in this study were designed to meticulously check the YOLOv8 and YOLOv11 series under various conditions. Each model was tested under different input sizes, anchor settings, and hyperparameters to optimize detection accuracy and computation efficiency. This included some key aspects, such as choosing the appropriate learning rate, batch size, and training epochs. The models of YOLOv8, starting from the nano version up to extra-large, were configured so that speed and precision go hand in hand. The YOLOv11 series introduced several advanced modifications, including improved backbone networks and optimized feature extraction layers, for enhanced performance. These configurations were tested using metrics such as mAP_{50}, mAP_{50-95}, precision, recall, and F1-score to ensure strong performance in a wide range of scenarios. Furthermore, the experiments were carried out on high-performance GPUs to ensure that the computational requirements of these models were met with ease.

The models trained were: YOLOv8n.pt, YOLOv8s.pt, YOLOv8m.pt, YOLOv8l.pt, YOLOv8x.pt, YOLO11n.pt, YOLO11s.pt, YOLO11m.pt, YOLO11l.pt, YOLO11x.pt

Since the SAR applications require the detection of small-sized objects, it becomes a disadvantage for the accuracy of a CNN. This created the need to add a separate, higher-resolution layer which can help differentiate closely spaced objects better. The P2 layer in the network was added to improve the detection accuracy [21]. This layer is incorporated in the .yaml file, which also specifies the train, val, and test source paths.

The above-listed models were run on the Kaggle platform [21]. Kaggle provided time-restricted access to NVIDIA-SMI 550.90.07 with dual GPU, the high-level structure of the same can be seen in Figure 3.

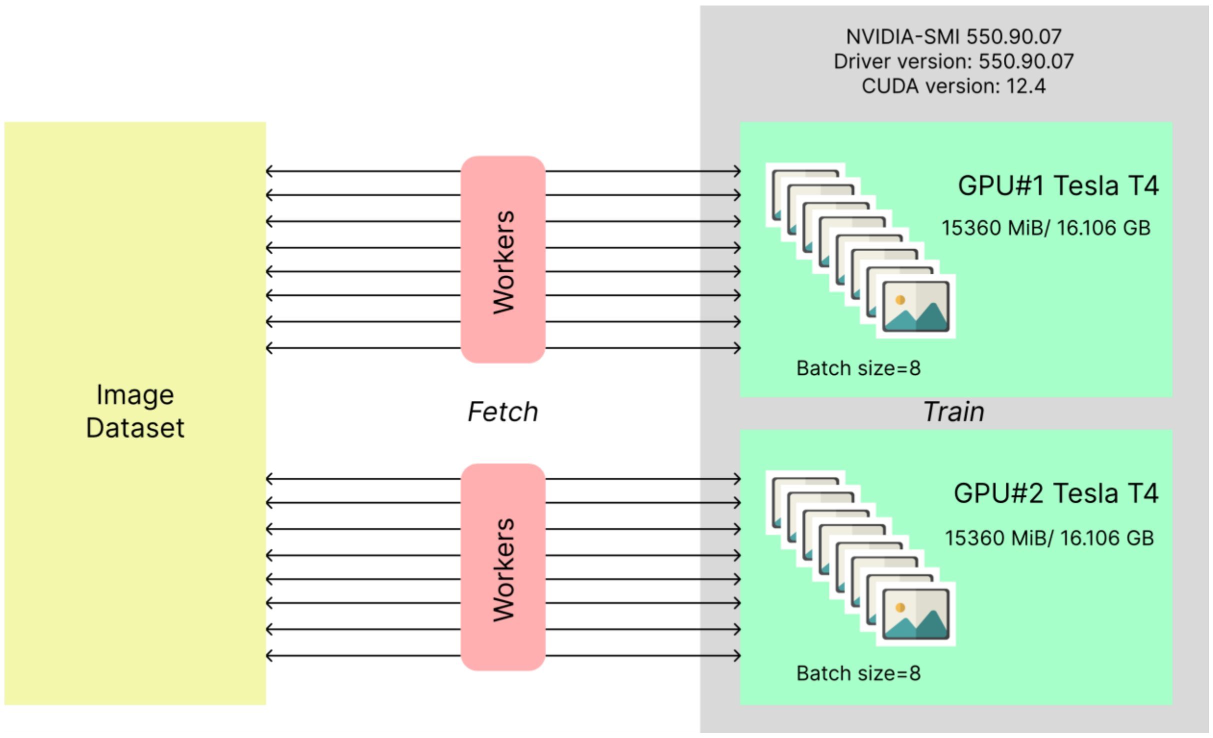


Figure 3. NVIDIA-SMI 550.90.07

The NVIDIA System Management Interface version 550.90.07 has been illustrated in Figure 3, wherein multiple workers fetch and preprocess the data ensuring that the I/O handling is efficient in order to avoid bottlenecks. The fetched data is then distributed in batches to two Tesla T4 GPUs. The parallelized training across the two GPUs accelerates computation by dividing the workload.

All models were trained at 50 epochs to balance the model accuracy and avoid overfitting. The batch size and workers were both set at 8, to help speed up data loading and processing while complying with the GPU memory limitations.

An IOU threshold of 0.7 was set to ensure that overlapping masks do not obscure objects, allowing better precision for images with densely packed elements.

The optimizer and momentum setting were set to ‘Auto’ during the training. The model was trained on ‘AdamW’ with a momentum of 0.9 and a learning rate of 0.000714 for all the models.

3.3 Experimental Set-up

The Unreal Engine (UE) version 4.2 [23] is utilized for developing the simulation setup. The Cesium plugin is used to generate Bing maps in the environment. 3D model of boats, swimmers, buoys, and lifesaving appliances is imported and placed in the environment to create a SAR scenario, as illustrated in Figure 4. The drone used in the UE simulation has been incorporated with two cameras, one for a third-person view and another pointing downwards, normal to the plane formed by the rotors of the drone. The latter is used to generate the feed which is fed to the object detection model. The drone in UE simulation has been made fully functional by adding drone physics and configuring controls by creating ‘blueprints’ in UE 4.2. The location of Marina Beach in Chennai (13.0500° N, 80.2824° E) is fed to the Cesium ion to simulate the SAR situation in real-world conditions. Figure: 5 shows the UE simulation running in the ‘Play’ mode.

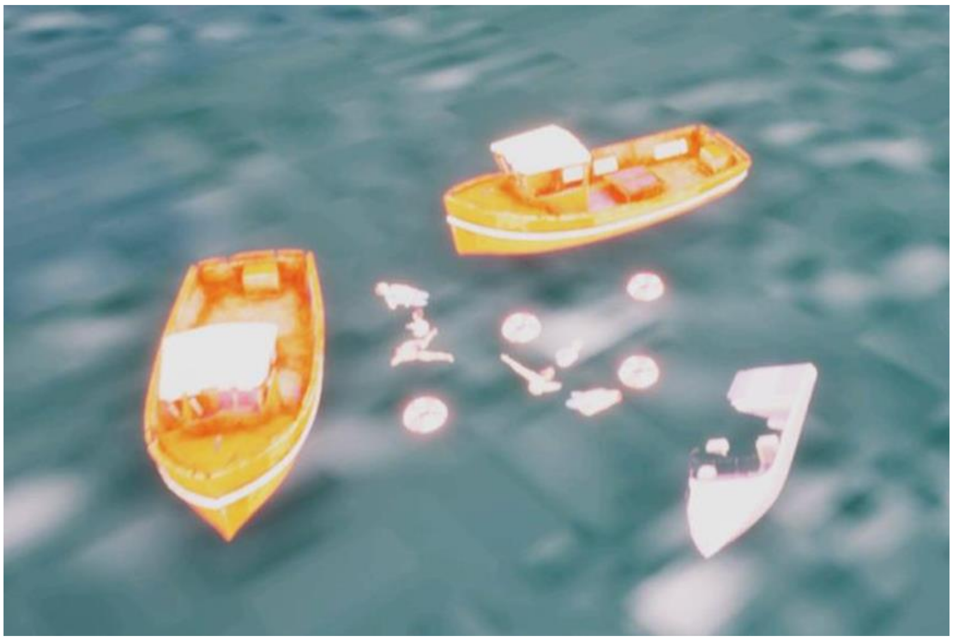


Figure 4. Import of 3D models for SAR scenario creation in UE 4.2

The simulated environment created with Unreal Engine is depicted in Figure 4, which portrays the realistic terrain, light, and texture. The simulated environment is designed as a testbed for validating drone operations, including navigation and obstacle avoidance for various conditions. This full-fidelity simulation mimics real-world scenarios, providing a controlled testbed for validating the drone's performance and robustness before deployment.

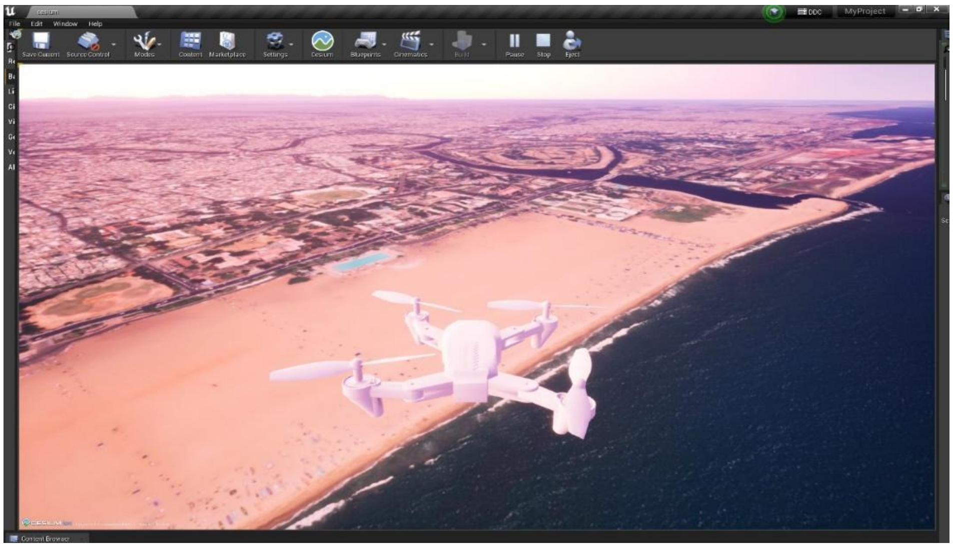


Figure 5. UE 4.2 simulation in Play mode

A simulated drone interacting with Unreal Engine obstacles is depicted in Figure 5. The figure highlights the drone's onboard sensors and algorithms and their ability to detect obstacles and navigate around them. This visualization underlines the importance of testing mechanisms like collision avoidance and path planning in a safe and realistic virtual setting.

The experimentation starts with the drone simulation running in UE 4.2. The video feed from the simulation is piped out to the selected object detection model. A prediction will be generated, with bounding boxes and a confidence score for the predicted objects in real time, which has been outlined in Figure 6.

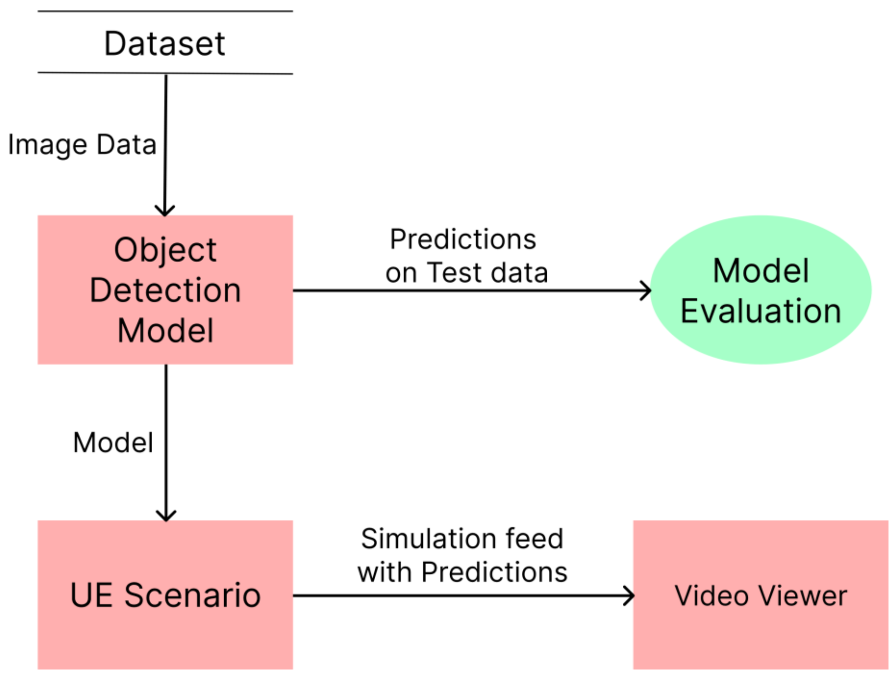


Figure 6. Data Flow Diagram for the planned execution

The sequential steps of the planned execution process are outlined in Figure 6. The process starts with data acquisition, which is the process of collecting raw inputs in the form of images or environmental parameters that form the foundation for further processing. It is then followed by the data pre-processing stage, which cleans, augments, and formats the raw data into something ready for analysis. The processed data is passed to the simulation or model execution phase where it is utilized to produce predictions or complete tasks like object detection or classification. The predictions are consolidated in the output generation stage to provide actionable results. The evaluation and feedback step then assesses the performance of the model through metrics like accuracy, precision, and recall. The obtained insights are therefore used to improve and enrich the system to make its performance iterative and reliable on execution. The flow structure assures clarity and efficiency throughout a process.

3.4 Evaluation Metrics

Across various annotated datasets employed by object detection challenges and the research community, the primary metric for evaluating detection accuracy is Average Precision (AP). To fully understand the variations of AP, it is essential to first familiarize ourselves with some fundamental terms commonly used in this context:

- True Positive (TP): A correct identification of a ground-truth bounding box.
- False Positive (FP): An incorrect identification, either detecting a non-existent object or misplacing the detection of an existing one.
- False Negative (FN): A failure to detect an actual ground-truth bounding box.

The count of True Negatives (TN) is irrelevant in object detection since the number of potential bounding that should not be identified within an image is infinite.

Precision and recall are important measures in machine learning that evaluate the performance of a model. Precision computes the correctness of positive predictions, representing the percentage of correct positive predictions. While recall determines how well the model recognizes all relevant instances.

A precision-recall curve (PR curve) is used for visualizing the relationship between Precision and Recall, across varying confidence thresholds assigned to the bounding boxes predicted by the detector.

In the 11-point interpolation method, the precision-recall curve is summarized by averaging the highest precision values at 11 evenly spaced recall levels: 0, 0.1, 0.2, ..., 1. The AP is calculated by considering the maximum interpolated precision, P_{interp}(R) at each recall level, rather than the observed precision, P(R).

$$AP_{all} = \sum_{n} (R_{n+1} - R_n) P_{interp}(R_{n+1})$$

where,

$$P_{interp}(R_{n+1}) = \max_{\tilde{R}:\tilde{R} \geq R_{n+1}} P(\tilde{R})$$

And $\tilde{R}$ denotes the mean of R values.

To measure the accuracy over all classes, the average of AP over all classes is taken, this metric is known as the mean average precision (mAP.)

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i$$

The average precision (AP) is a per-class measure while mAP is the average of APs across all the classes. Thus, mAP is a robust evaluation metric that considers multiple queries.

A confusion matrix is a tabular representation of a model’s performance. It compares the predicted labels to the actual labels and provides detailed insights into the model's performance in each class. For object detection, the confusion matrix is typically used for evaluating the classification of detected objects. Figure: 7 shows the structure of a confusion matrix.

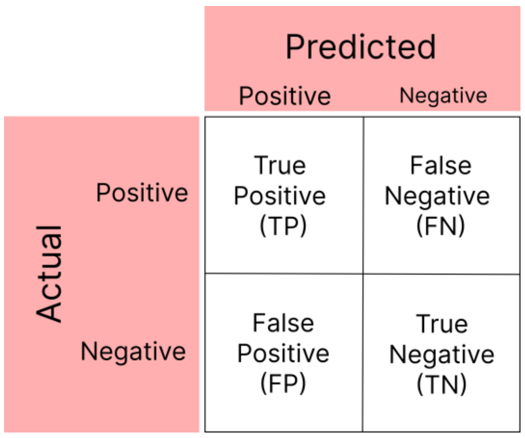


Figure 7. Structure of a confusion matrix

The structure of a confusion matrix has been depicted in Figure 7, which serves as an important evaluation tool in classification tasks. It consists of four key components: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). The matrix helps analyze model performance, providing insights into accuracy, precision, recall, and overall error rates for the trained model.

Formulae of evaluation metrics like Precision, Recall, Negative Predictive Value, Specificity, and Accuracy, are as follows:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Negative\ Predicted\ Value = \frac{TN}{TN + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

The F1 score is a metric used to measure the performance of a machine learning model. It can be calculated by combining the precision and recall of the model. The F1 score can be calculated in the following way:

$$F1\ score = \frac{2 * Precision * Recall}{Precision + Recall}$$

4. Result and Discussion

4.1 Quantitative Performance Analysis

Table 4. Performance comparison of the trained models across all the classes (scale of 0-1)

Model	GFLOPs	Precision	Recall	mAP$_{50}$	mAP$_{50-95}$	F1 score	Accuracy
YOLOv8n	8.1	0.818	0.601	0.621	0.373	0.693	0.488
YOLOv8s	28.4	0.845	0.633	0.660	0.408	0.724	0.542
YOLOv8m	78.7	0.691	0.638	0.666	0.412	0.663	0.571
YOLOv8l	164.8	0.734	**0.653**	0.680	**0.432**	0.691	**0.576**
YOLOv8x	257.4	0.854	0.649	0.687	0.429	**0.737**	**0.575**
YOLOv11n	6.3	0.810	0.592	0.611	0.369	0.684	0.481
YOLOv11s	21.3	0.855	0.628	0.668	0.410	0.724	0.546
YOLOv11m	67.7	0.848	0.632	**0.692**	0.422	0.724	**0.571**
YOLOv11l	86.6	0.844	0.647	0.688	0.422	0.732	0.569
YOLOv11x	194.4	**0.861**	0.629	0.677	0.426	0.727	0.562

In analyzing the performance metrics across the YOLOv8 and YOLOv11 models, several key insights emerge, which have been documented in Table 4. YOLOv8 models span from 8.1 GFLOPs for YOLOv8n to 257.4 GFLOPs for YOLOv8x, while YOLOv11 models are generally more computationally efficient, ranging from 6.3 GFLOPs for YOLOv11n to 194.4 GFLOPs for YOLOv11x. Smaller models in both series consume significantly fewer computational resources, with YOLOv11 models typically requiring less. In terms of precision, YOLOv11 models slightly outperform YOLOv8 models, varying from 0.691 for YOLOv8m, to 0.861 for YOLOv11x. The most precise models are YOLOv11x (0.861), YOLOv8x is the second with 0.854, and then YOLOv11s with 0.855. The recall is relatively level across models, ranging from a minimum of 0.592 for YOLOv11n to a maximum of 0.653 for YOLOv8l, however, most models are still within the range of between 0.6 to 0.65, revealing similar object detection capabilities between models. For mAP50, YOLOv11m has the lead at 0.692 and YOLOv8x follows closely at 0.687, showing very good performance for both series. For mAP50-95, values are lower, ranging from 0.369 for YOLOv11n up to 0.432 for YOLOv8l, indicating diminished performance at different IoU thresholds. The F1 scores, which balance precision with recall, are between 0.663 for YOLOv8m and 0.737 for YOLOv8x. YOLOv8x and YOLOv11x both perform strongly. Accuracy values cluster between 0.481 for YOLOv11n and 0.576 for YOLOv8l, with most models in the range of 0.55 to 0.57, thereby having equivalent overall performance. In summary, although YOLOv11 models are computationally more efficient with slightly higher precision, YOLOv8 models perform competitively in particular concerning F1 score and mAP50, so both series can be considered good options based on the requirements for efficiency, precision, and performance.

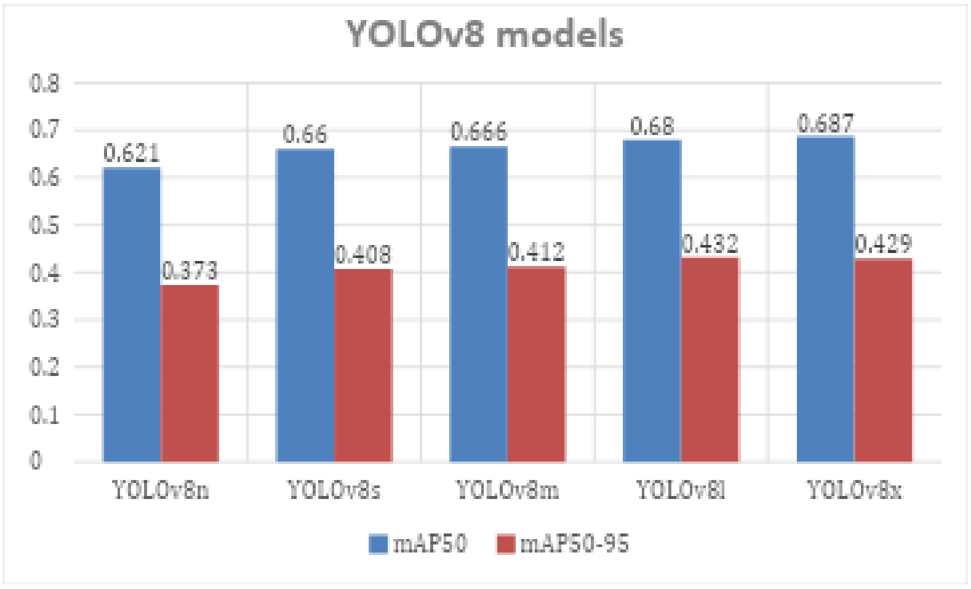


Figure 8. Column chart for comparing mAP_{50} and $mAP_{50\text{-}95}$ values of YOLOv8 models

Comparison of mAP_{50} (mean average precision at IoU threshold 0.5) and $mAP_{50–95}$ (mean average precision averaged over IoU thresholds 0.5 to 0.95) for YOLOv8 models: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x are shown in Figure 8, which indicates that, with higher model complexity, both metrics grow, and the highest scores are achieved by YOLOv8x, namely mAP_{50}: 0.687, $mAP_{50–95}$: 0.429. This indicates that the more complex larger models would perform better on detection precision across varying IoU thresholds. However, the model selection based on complexity and efficiency must be well-balanced.

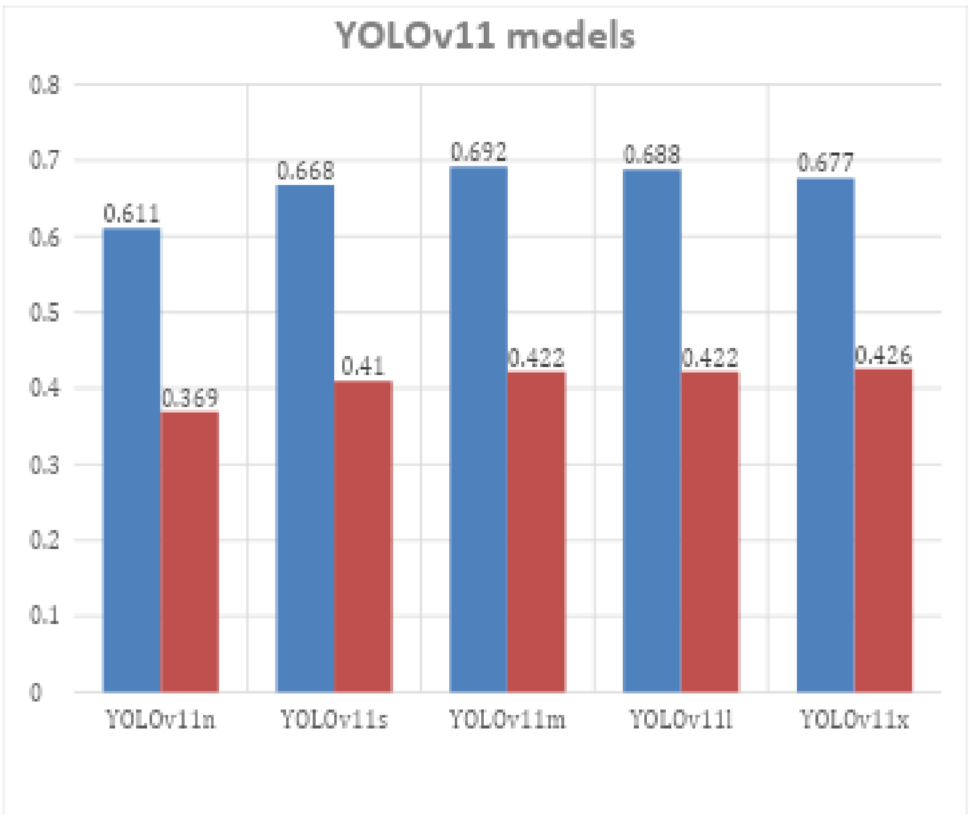


Figure 9. Column chart for comparing mAP_{50} and $mAP_{50\text{-}95}$ values of YOLOv11 models

The mAP50 and mAP50-95 values for the different YOLOv11 models: nano, small, medium, large, and extra-large have been depicted in Figure 9, which indicates a non-linear trend in the performance of the model. The mAP50 value peaks at 0.692 for the YOLOv11m (medium) model and then declines toward the YOLOv11x (extra-large) model. This therefore suggests that a medium-sized model performs best on this particular task of maritime search and rescue object detection, with bigger models not necessarily resulting in a more accurate output.

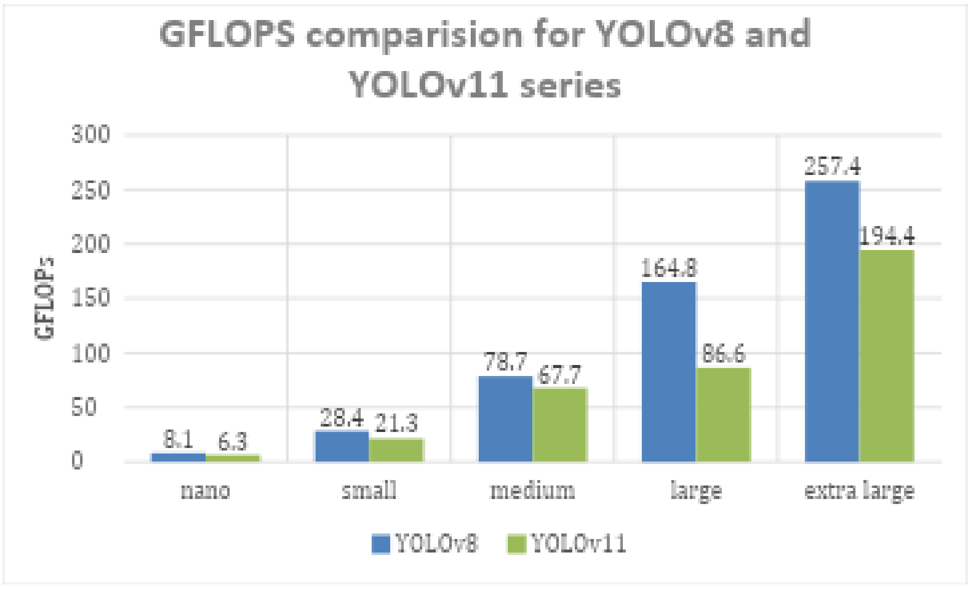


Figure 10. Comparing GFLOP values for different models across YOLOv8 and YOLOv11 series

A detailed comparison of GFLOP (Giga Floating Point Operations) for different models in the YOLOv8 and YOLOv11 series is depicted in Figure 10. It helps analyze which model is computationally intensive and resource-demanding. Models in the YOLOv8 series are considered as balanced between efficiency and performance, while models in the YOLOv11 series, with higher GFLOP values, are more concentrated on increasing accuracy and capability to perform tasks that require such precision. This comparison is critical for choosing the best model in terms of computational resources and task requirements.

The comparison in Figure 10 reveals that YOLOv11 models are computationally lightweight while offering competitive performance to YOLOv8 models.

4.2 Performance evaluation of top 3 performing models

The class-wise performances of YOLOv11m (Accuracy: 0.571), YOLOv8l (Accuracy: 0.576), and YOLOv8x (Accuracy: 0.575) are tabulated in Table 5 (for YOLOv11m), Table 6 (for YOLOv8l), and Table 7 (YOLOv8x).

Table 5. Class-wise performance for YOLOv11m

Class	**Precision**	**Recall**	**mAP$_{50}$**	**mAP$_{50-95}$**
boat	0.915	0.920	0.931	0.689
buoy	0.860	0.769	0.807	0.526
jetski	0.761	0.893	0.914	0.586
life_saving_appliances	1.000	0.000	0.181	0.063
swimmer	0.707	0.577	0.628	0.246

Table 6. Class-wise performance for YOLOv8l

Class	Precision	Recall	mAP$_{50}$	mAP$_{50\text{-}95}$
boat	0.918	0.905	0.934	0.719
buoy	0.847	0.788	0.804	0.524
jetski	0.837	0.947	0.953	0.645
life_saving_appliances	0.398	0.009	0.008	0.025
swimmer	0.671	0.618	0.629	0.249

Table 7. Class-wise performance for YOLOv8x

Class	Precision	Recall	mAP$_{50}$	mAP$_{50\text{-}95}$
boat	0.896	0.922	0.950	0.711
buoy	0.873	0.781	0.807	0.526
jetski	0.818	0.935	0.948	0.629
life_saving_appliances	1.000	0.000	0.103	0.034
swimmer	0.681	0.608	0.628	0.246

The poor performance of the ‘life_saving_appliances’ class can be understood by Table 8, which shows the number of instances for every class in the validation dataset.

Table 8. Class-wise performance for YOLOv8x

Class	Images	Instances
boat	1209	2413
buoy	806	934
jetski	768	769
life_saving_appliances	592	884
swimmer	1902	11151

Intriguing class-wise performance metrics have emerged based on the object detection of YOLOv11m, YOLOv8l, and YOLOv8x on various classes of maritime objects. All three models performed seemingly well when detecting boats and jetskis with precision and recall more than 0.7. Boats are found to achieve the highest mAP50 performance at 0.931 through 0.950 on all of the models.

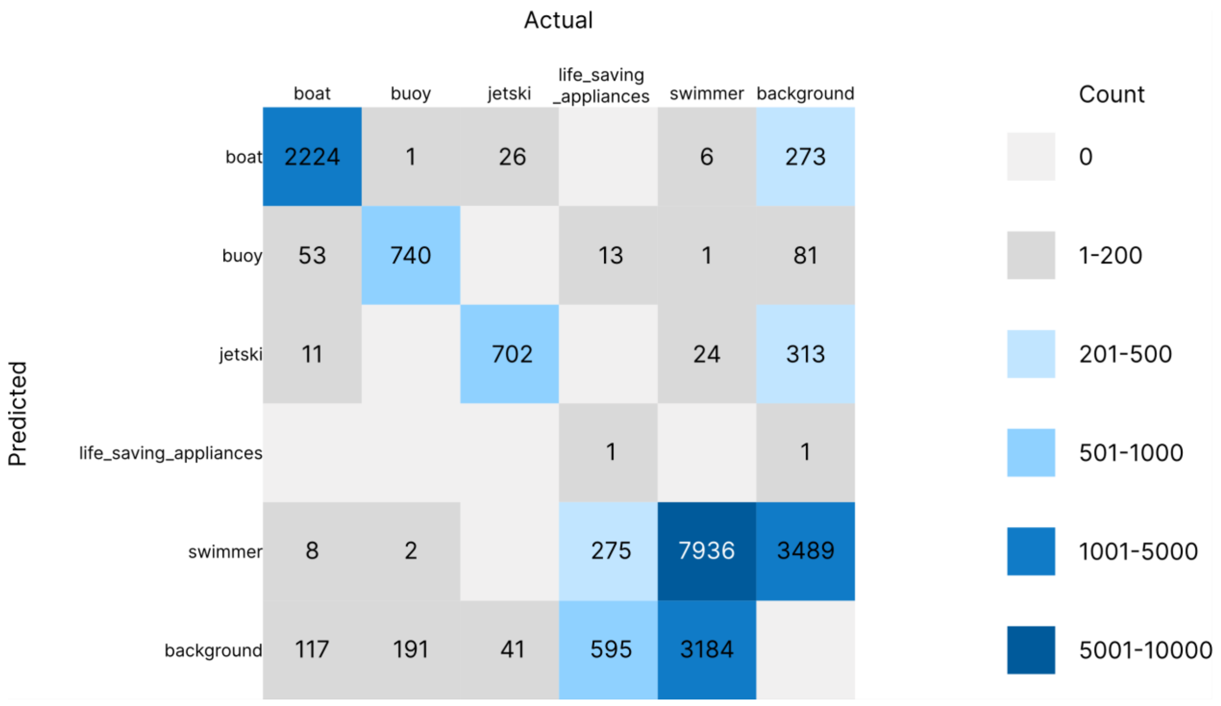
However, in the 'life_saving_appliances' class, a major flaw surfaced. Despite 884 examples of this class as per Table 8, the performance of all the models collapsed, with YOLOv11m and YOLOv8x having perfect precision and zero recall, while the latter had very limited capability of detecting any of these. Such terrible performance can be attributed to them being small objects and having a low representation in the validation dataset.

The swimmer class also indicated moderate performance as precision and recall float close to 0.6-0.7 which shows an area for enhancement in target detection towards a human figure in maritime.

The small size of objects belonging to ‘life_saving_appliances,’ coupled with the least number of images in the validation dataset has resulted in poor scores. A suggested approach to solve this problem would be to re-arrange the data between the three data subsets (train, val, and test).

4.3 Confusion Matrix and Precision-Recall Curve of top 3 performing models

A confusion matrix is a 2-D matrix used to evaluate the performance of a model by showing the number of correctly and wrongly classified data. The confusion matrices for YOLOv11m, YOLOv8l, and YOLOv8x, have been depicted in figures 11a, 11b, and 11c respectively. The confusion matrices for these three models can be studied to obtain the accuracy of each model.

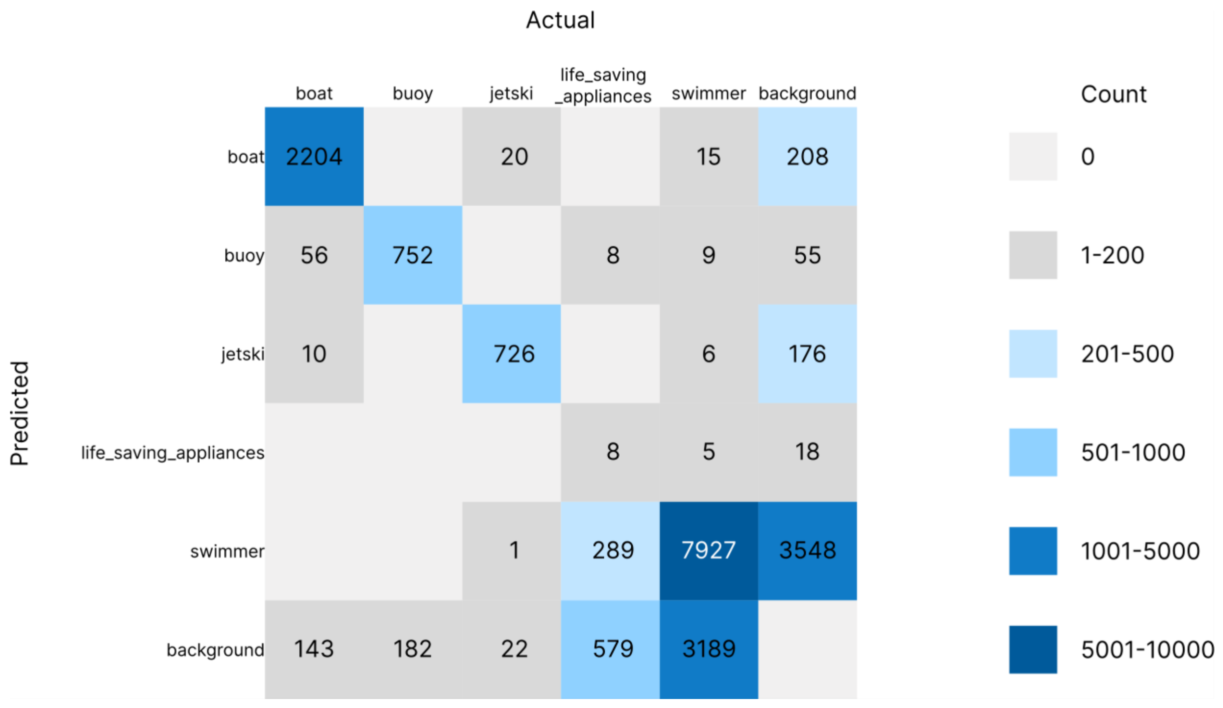


Actual

Predicted	boat	buoy	jetski	life_saving _appliances	swimmer	background
boat	2224	1	26		6	273
buoy	53	740		13	1	81
jetski	11		702		24	313
life_saving_appliances				1		1
swimmer	8	2		275	7936	3489
background	117	191	41	595	3184	

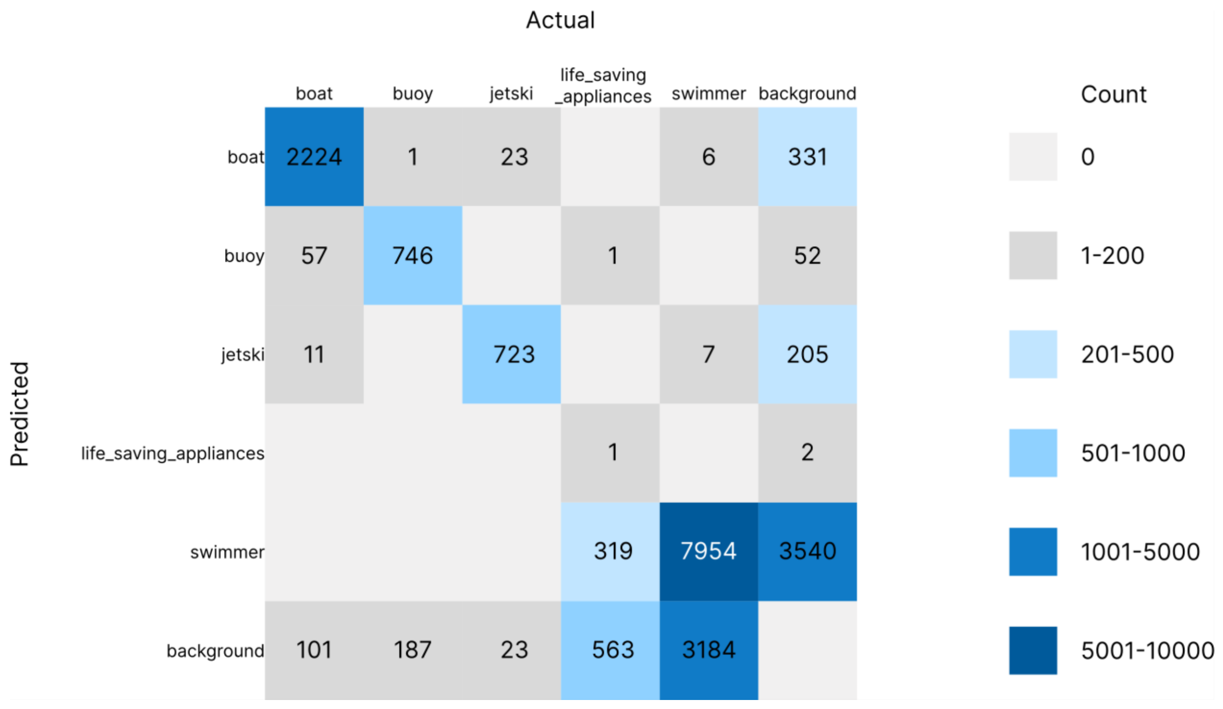
Count	
⬜	0
⬜	1-200
🟦	201-500
🟦	501-1000
🟦	1001-5000
🟦	5001-10000

Figure 11a. Confusion matrix for YOLOv11m



	boat	buoy	jetski	life_saving _appliances	swimmer	background
boat	2204		20		15	208
buoy	56	752		8	9	55
jetski	10		726		6	176
life_saving_appliances				8	5	18
swimmer			1	289	7927	3548
background	143	182	22	579	3189	

Figure 11b. Confusion matrix for YOLOv8l



	boat	buoy	jetski	life_saving _appliances	swimmer	background
boat	2224	1	23		6	331
buoy	57	746		1		52
jetski	11		723		7	205
life_saving_appliances				1		2
swimmer				319	7954	3540
background	101	187	23	563	3184	

Figure 11c. Confusion matrix for YOLOv8x

The confusion matrices of YOLOv11m, YOLOv8l, and YOLOv8x represent patterns in maritime object detection. Boats and jetskis show the highest correct classification for all models and are quite robust at larger objects. Swimmers are the hardest to misclassify, and they show the most confusion between classes. The class of life_saving_appliances is the most challenging and has the least number of correct identifications.

The diagonal elements corresponding to the correct predictions are surprisingly alike in the distributions across the three models. This could imply that, although they differ slightly, the models approach object detection in maritime scenes with comparable strategies and limitations.

Table 9 displays the accuracy obtained by the three models; YOLOv11m, YOLOv8l, and YOLOv8x. The accuracy is computed by analyzing the confusion matrices.

Table 9. Accuracy of the top three performing models

Model	**Accuracy**
YOLOv11m	0.571
YOLOv8l	0.576
YOLOv8x	0.575

The PR-curves of YOLOv11m, YOLOv8l, and YOLOv8x reveal the trade-offs between the precision and recall for different confidence scores, which have been depicted in Figures 12a, 12b, and 12c. Every curve follows a generic trend; the precision dips while the recall increases typical phenomenon for object detection models. The area under the curve is relatively balanced across the three models, which makes them of comparable performance. The curves indicate that for lower values of recall, the models have precision, then gradually decline in their effort to detect more objects. There are minute differences in the shapes of the curves, which shows the difference in the approaches of the detection of maritime scenarios among the models.

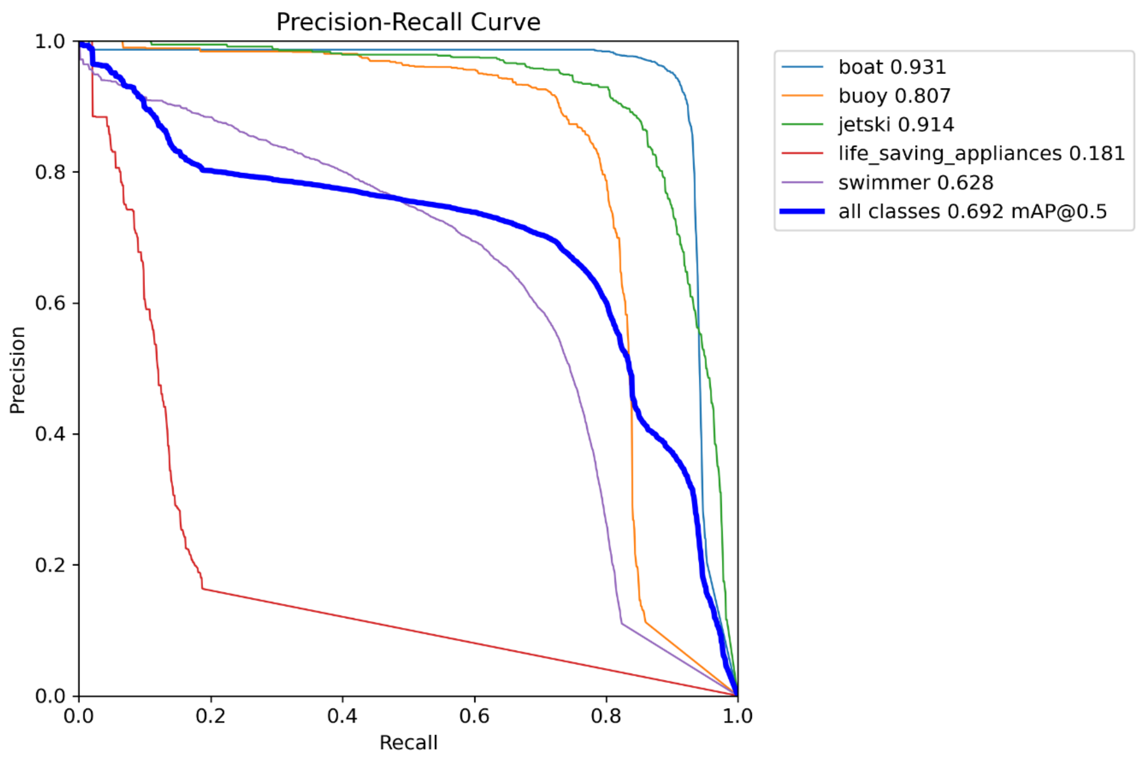


Figure 12a. Precision-Recall curve for YOLOv11m

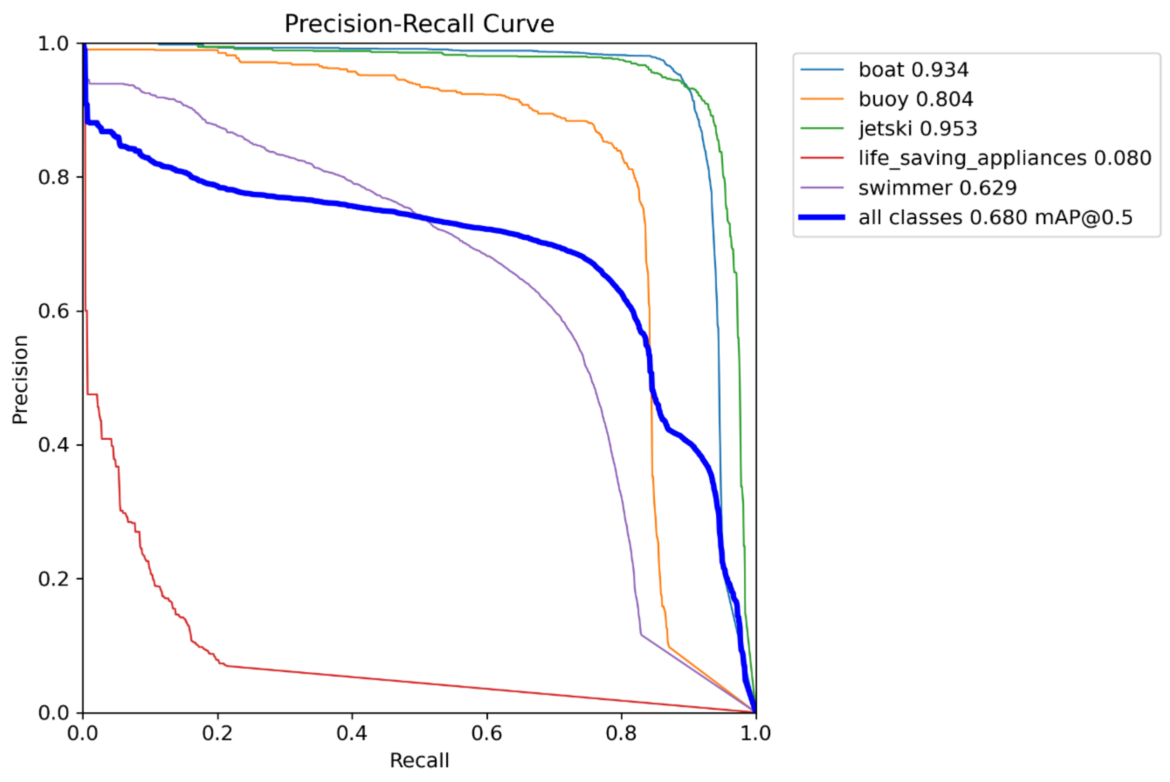


Figure 12b. Precision-Recall curve for YOLOv8l

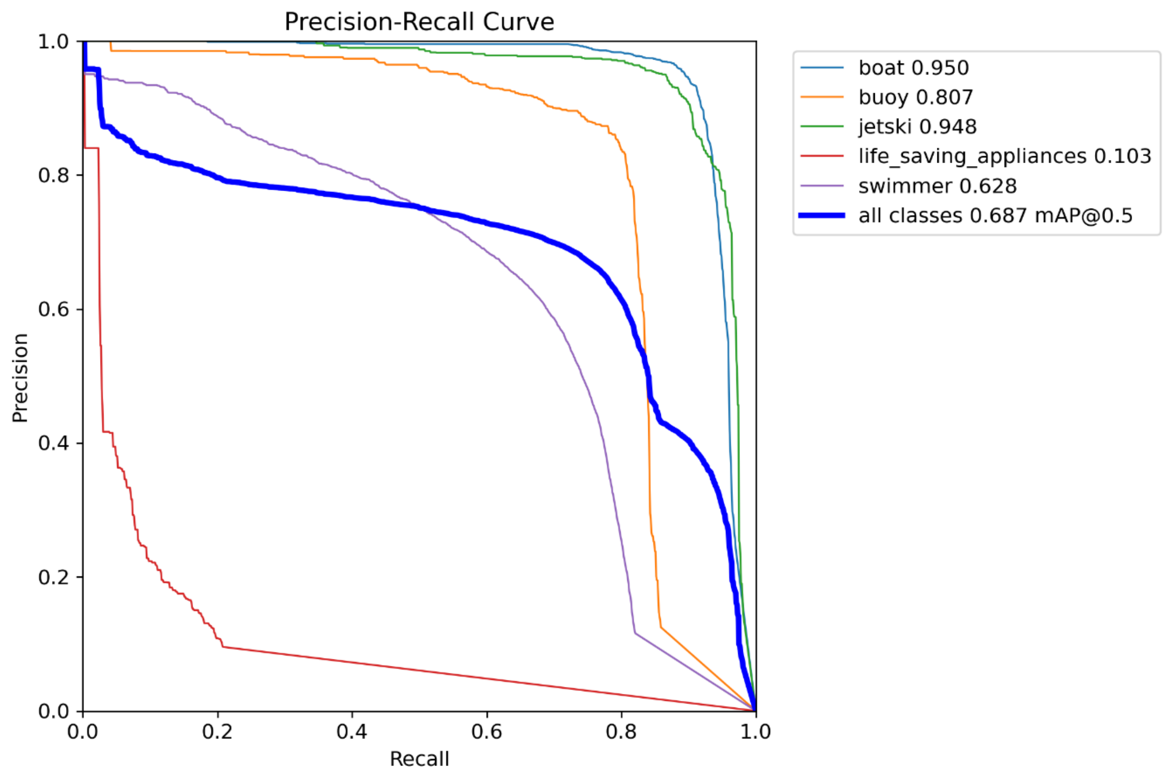


Figure 12c. Precision-Recall curve for YOLOv8x

4.4 Representative Outputs of the Detection Models

Predicted results from the test data on the YOLOv8x model are shown in Figures 13a, 13b, 13c, 13d, and 13e.

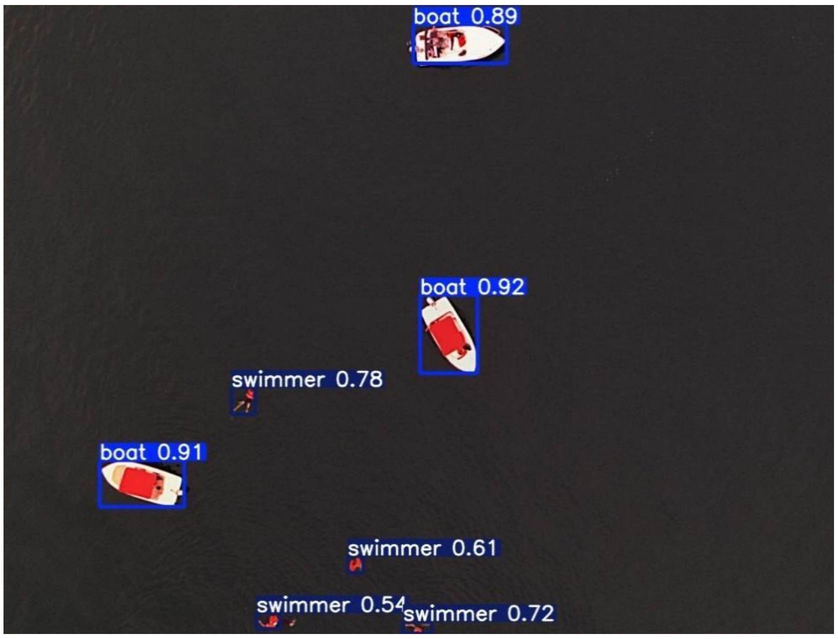


Figure 13a. Detection of boats and swimmers

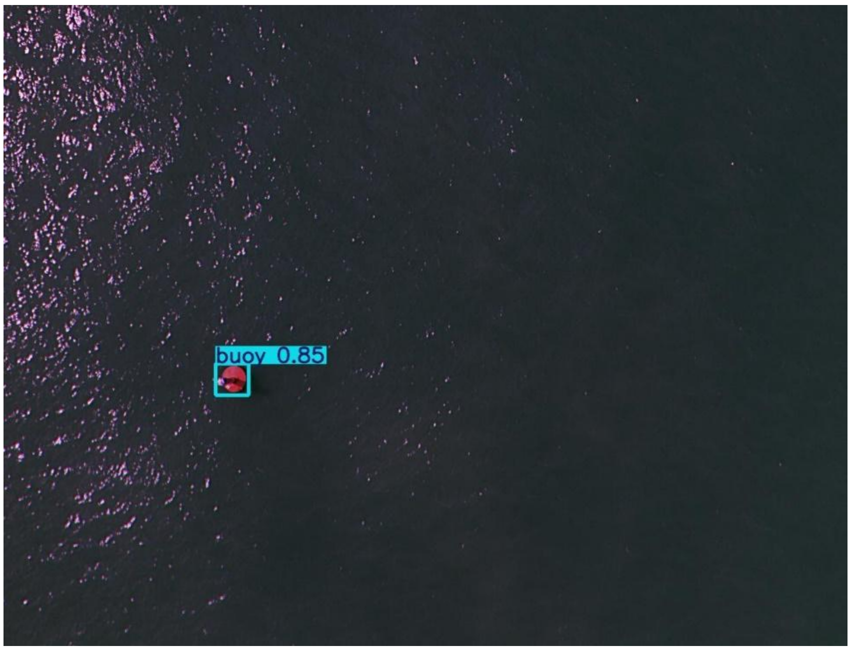


Figure 13b. Detection of a buoy

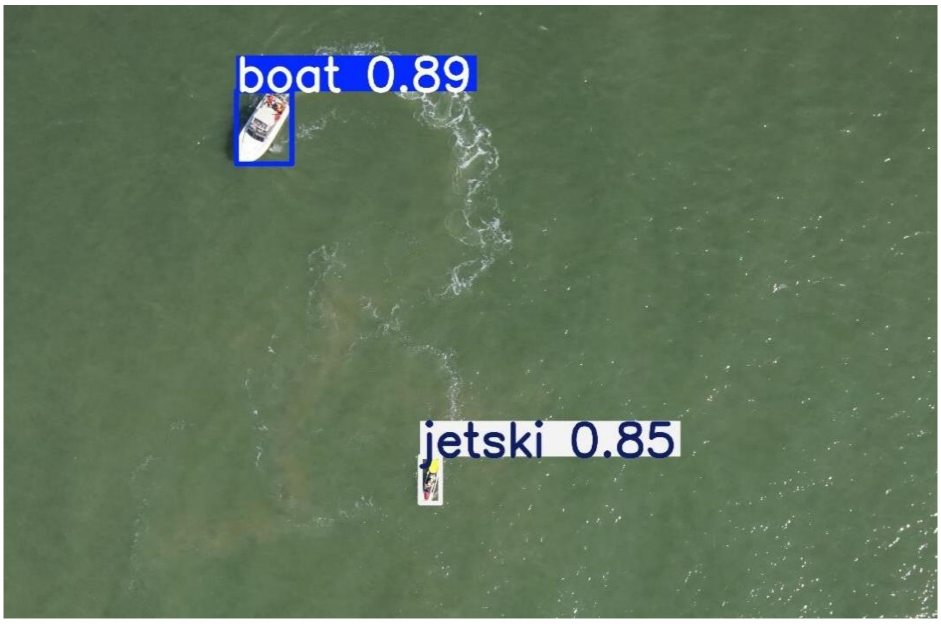


Figure 13c. Detection of a boat and a jetski

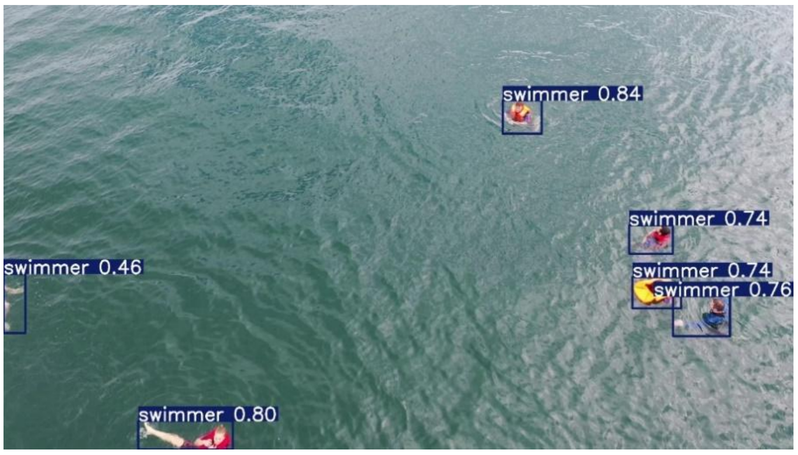


Figure 13d. Detection of swimmers

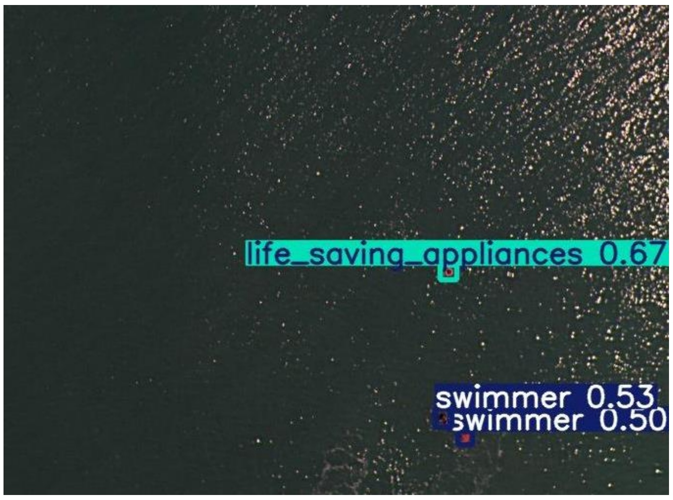


Figure 13e. Detection of swimmers and life_saving_appliances

The collective comparison of outputs or performances over multiple metrics or visual representations identified as 13a, 13b, 13c, 13d, and 13e reveals variations of results or outcomes under specific conditions, including changes to model parameters, input data, or environments. Analyses of the subfigures reveal a variety of performance differences based on accuracy, precision, and clarity. For instance, one of the subfigures may represent better object detection accuracy and another may show the weaknesses in certain classes or conditions. These comparisons altogether reflect the consistency of the model, strength, and areas for improvement for robust performance.

With boats, swimmers, etc. a test scenario was created in UE to simulate the SAR environment. A real-time simulation in Unreal Engine was passed to the CNN models for generating predictions. Screen captures of the video feed with generated predictions are shown in Figures 14a, 14b, and 14c.

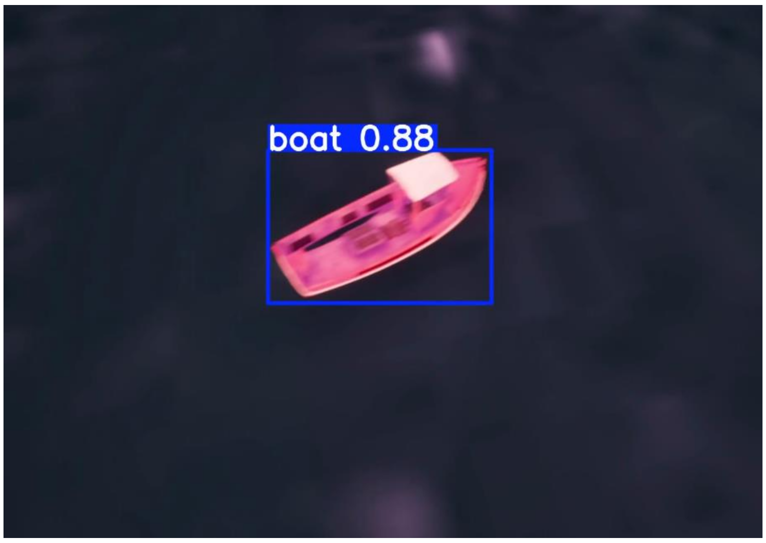


Figure 14a. Screen-clipping of the simulation with predictions (1)

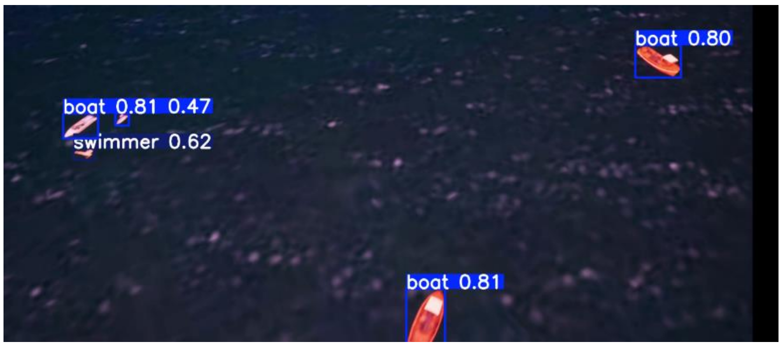


Figure 14b. Screen-clipping of the simulation with predictions (2)

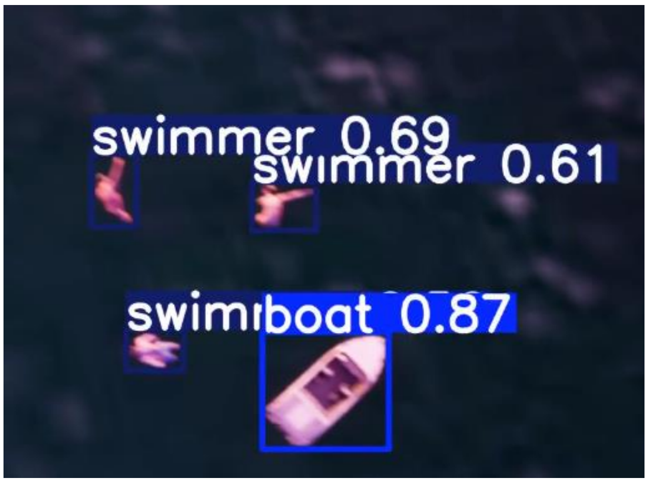


Figure 14c. Screen-clipping of the simulation with predictions (3)

The model's outputs or results under various conditions have been analyzed in Figures 14a, 14b, and 14c. Each subfigure has a different aspect of performance that it focuses on, including detection accuracy, computational efficiency, and object classification across the varying scenarios. For example, Figure 14(a) could emphasize higher precision while 14(b) has better recall, and 14(c) could reveal problems with specific classes or environmental factors. Collectively, these numbers allow for a comprehensive analysis of the model's performance, thereby giving insights into its strengths, weaknesses, and overall consistency. This comparison is important to identify areas of further optimization and robust results in various test cases.

4.5 Analysis of Model Performance Relative to Size

The performance analysis gives a clear correlation between model size (GFLOPs) and accuracy metrics. With YOLOv11x showing slightly higher precision (0.861) compared to YOLOv8x (0.854), larger models like YOLOv8x (257.4 GFLOPs) and YOLOv11x (194.4 GFLOPs) have achieved a higher precision and recall. However, this advantage costs higher computational requirements, which could be a limiting factor for their deployment on edge devices.

Smaller models such as YOLOv8n and YOLOv11n have scored lower precision and recall but have also consumed much fewer GFLOPs. This makes them suitable for applications requiring faster inference with constrained resources. Interestingly, YOLOv11n, despite having lower GFLOPs (6.3), showed competitive precision (0.810) and recall (0.592), comparable to YOLOv8n.

4.6 Impact of Model Size on Rescue Operations Efficiency

Speed and accuracy are critical for rescue operations. Models like YOLOv11m (67.7 GFLOPs) struck a balance by offering high precision (0.848) and recall (0.632) while maintaining a relatively moderate computational load. This makes them ideal for real-time detection tasks where deploying high-end hardware may not be feasible. While YOLOv8x and YOLOv11x perform slightly better in terms of precision, their heavier computational demands may reduce the overall efficiency of real-time rescue operations.

4.7 Comparison of YOLOv8 and YOLOv11 models

Compared to YOLOv8, YOLOv11 models show slight improvements in precision across all variants, indicating enhanced detection capabilities. For example, YOLOv11s achieved a precision of 0.855, surpassing YOLOv8s's 0.845, with a similar GFLOPs range. Similarly, YOLOv11m demonstrated a higher mAP50-95 (0.422) compared to YOLOv8m's 0.412, showcasing its effectiveness in detecting objects of varying scales. These advancements highlight the potential of YOLOv11 models as better alternatives to YOLOv8 in disaster response scenarios.

4.8 Comparison to Other Existing Models

A paper titled 'Modular YOLOv8 optimization for real-time UAV maritime rescue object detection' [26] released in 2024, presents a comparative study of object detection models utilizing the SeaDronesSee dataset [20]. Table: 10 compares the results from the referred research to the models proposed in this paper.

Table 10. Accuracy of the top three performing models

Model	**mAP_{50} in Referred Research [20] (in %)**	**mAP_{50} in this research (in %)**
YOLOv8n	37.9	62.1
YOLOv8s	39.7	66.0
YOLOv8-PA-s	52.0	-
YOLOv8-PA-CBAM-s	59.1	-
YOLOv8-PA-STP3-s	52.4	-
YOLOv8-PA-GAM-s	57.5	-
YOLOv8-PA-STP3 -CBAM-s	53.8	-
YOLOv8-PA-STP3-GAM-CBAM-s	54.1	-
YOLOv8m	44.7	66.6
YOLOv8l	42.3	68.0
YOLOv8x	46.1	68.7

Every model in the YOLOv8 series proposed in this study has performed better (mAP_{50} score) against every model presented in [20]. Another model that achieved the best result in [20] was Faster R-CNN with ResNeXt101-FPN with a mAP_{50} score of 62.8%.

Thus, the models proposed in this paper have achieved ground-breaking results in comparison to recent models, utilizing similar data.

5. Conclusion

5.1 Summary of Key Findings

The study concludes that YOLOv11 models outperform YOLOv8 in precision and recall across most variants. YOLOv11x achieved the highest precision (0.861), while YOLOv11m offered an optimal trade-off between accuracy and computational efficiency, making it well-suited for real-time applications in rescue operations. The comparison further underscores the scalability of YOLO models, allowing selection based on the constraints and requirements of specific use cases.

5.2 Limitations of the Study

• The analysis relied on specific benchmark datasets, which may not cover all scenarios encountered during real-world rescue operations.

• The computational cost evaluation did not account for deployment on various hardware configurations, potentially impacting the models' adaptability across different platforms.

• Object detection performance in poor visibility or adverse weather conditions was not explicitly tested.

5.3 Recommendations for Future Research on YOLOv8 and YOLOv11 in Rescue Operations

Dataset Expansion: Develop and evaluate models on diverse, real-world rescue operation datasets, including those with poor lighting and adverse weather.

Hardware Optimization: Optimize YOLOv11 models for deployment on edge devices with limited computational power.

Scenario Testing: Conduct in-depth analysis of dynamic environments, such as moving objects or obstacles in water-based rescue scenarios.

Transfer-Learning Approach: Pre-train the model on classes like ‘life_saving_appliances’ and import their weights to improve the performance in specific classes. This approach is also called the transfer-learning approach in machine learning.

Energy Efficiency: Investigate power consumption and model efficiency to ensure sustainable deployment in prolonged rescue missions.

Improvements in Drone Simulation Scenario Creation: The simulation scenario could be expanded with a greater variety of objects imported into the scene like boats, jet skis, life-saving appliances, etc.

CRediT authorship contribution statement

Spandan Kiran Vaidya: Investigation; Methodology; Software; Validation; Writing - original draft. **Sakthivel Velusamy:** Conceptualization; Methodology; Resources; Supervision. **Parit Gupta:** Data curation; Writing - results. **Rupankar Majumdar:** Software; Simulation - scenario creation; Writing – results. **Jeevan S:** Writing - review and editing; Simulation – object creation.

Declaration of competing interest

The authors assert that they have no identifiable conflicting financial interests or personal associations that might have influenced the study's outcomes. Consequently, they affirm that no contradictory interests are present.

Data availability

Data will be made available on request.

Acknowledgments

The authors thank Vellore Institute of Technology Chennai for the provision of resources for computation and analysis.

References