# UNDERWATER PLASTIC WASTE DETECTION WITH YOLO AND VISION TRANSFORMER MODELS

JONATHAN BRUCE CÁRDENAS RONDOÑO
https://orcid.org/0009-0009-6169-808X
jonathan.c.rondono@gmail.com
Universidad de Lima, Perú

NERS ARMANDO VASQUEZ ESPINOZA
https://orcid.org/0009-0007-5875-7638
nersarmandovasquezespinoza@gmail.com
Universidad de Lima, Perú

EDWIN JONATHAN ESCOBEDO CÁRDENAS
https://orcid.org/0000-0003-2034-513X
eescobed@ulima.edu.pe
Universidad de Lima, Perú

Received: April 21, 2025/ Accepted: June 4, 2025
doi: https://doi.org/10.26439/interfases2025.n021.7868

ABSTRACT. This study addresses the global issue of marine pollution, with a particular focus on plastic bag contamination, by leveraging real-time object detection techniques powered by deep learning algorithms. A detailed comparison was carried out between the YOLOv8, YOLO-NAS, and RT-DETR models to assess their effectiveness in detecting plastic waste in underwater environments. The methodology encompassed several key stages, including data preprocessing, model implementation, and training through transfer learning. Evaluation was conducted using a simulated video environment, followed by an in-depth comparison of the results. Performance assessment was based on critical metrics such as mean average precision (mAP), recall, and inference time. The YOLOv8 model achieved an mAP50 of 0.921 on the validation dataset, along with a recall of 0.829 and an inference time of 14.1 milliseconds. The YOLO-NAS model, by contrast, reached an mAP50 of 0.813, a higher recall of 0.903, and an inference time of 17.8 milliseconds. The RT-DETR model obtained an mAP50 of 0.887, a recall of 0.819, and an inference time of 15.9 milliseconds. Notably, despite not having the highest mAP, the RT-DETR model demonstrated superior detection performance when deployed in real-world underwater conditions, highlighting its robustness and potential for practical environmental monitoring.

KEYWORDS: object detection / deep learning / plastic waste / object detection model / underwater images

## DETECCIÓN DE RESIDUOS PLÁSTICOS SUBMARINOS CON MODELOS YOLO Y VISION TRANSFORMER

RESUMEN. Este estudio aborda el problema global de la contaminación marina, con un enfoque particular en la contaminación por bolsas de plástico, aprovechando técnicas de detección de objetos en tiempo real impulsadas por algoritmos de aprendizaje profundo. Se realizó una comparación detallada entre los modelos YOLO v8, YOLO-NAS y RT-DETR para evaluar su efectividad en la detección de desechos plásticos en entornos submarinos. La metodología abarcó varias etapas clave, incluyendo el preprocesamiento de datos, la implementación del modelo y el entrenamiento utilizando aprendizaje por transferencia. La evaluación se llevó a cabo a través de un entorno de video simulado, seguido de una comparación exhaustiva de los resultados. La evaluación del rendimiento se basó en métricas críticas como la precisión promedio (mAP), el *recall* y el tiempo de inferencia. El modelo YOLO v8 alcanzó un mAP50 de 0,921 en el conjunto de validación, con un *recall* de 0,829 y un tiempo de inferencia de 14,1 milisegundos. El modelo YOLO-NAS, en contraste, alcanzó un mAP50 de 0,813, un *recall* más alto de 0,903 y un tiempo de inferencia de 17,8 milisegundos. El modelo RT-DETR obtuvo un mAP de 0,887, un *recall* de 0,819 y un tiempo de inferencia de 15,9 milisegundos. Notablemente, a pesar de no tener el mAP más alto, el modelo RT-DETR demostró un rendimiento superior en la detección cuando se implementó en condiciones submarinas reales, destacando su robustez y potencial para aplicaciones prácticas de monitoreo ambiental.

PALABRAS CLAVE: detección de objetos / aprendizaje profundo / residuos plásticos / modelos de detección de objetos / imágenes submarinas

## INTRODUCTION

The National Institute of Statistics and Informatics (2022) identifies water pollution as a critical environmental issue, with plastic waste posing a growing threat to marine ecosystems. The increasing volume of plastic debris—such as bottles and bags—accumulating on beaches and seabeds contributes significantly to long-term ecological degradation. A major challenge in addressing this problem lies in the inaccessibility and high cost of monitoring underwater environments. Autonomous underwater vehicles (AUVs) present a promising solution, but their success depends on the integration of accurate object detection systems capable of identifying submerged plastic waste.

Deep learning has emerged as the leading approach for underwater object detection, offering real-time performance and high accuracy. However, the complexity of underwater environments—including low visibility, variable lighting, and cluttered seabeds—hampers detection accuracy (Dhana Lakshmi & Santhanam, 2020). Compounding this issue is the lack of diverse and comprehensive datasets, which limits model generalizability in real-world conditions (Hong et al., 2020; Panwar et al., 2020).

In response to these challenges, two main categories of object detection algorithms have been explored: one-stage and two-stage models. One-stage models, such as You Only Look Once (YOLO), are known for their fast detection speed but tend to sacrifice some accuracy. Conversely, two-stage models offer greater precision but at the cost of computational efficiency (Conley et al., 2022; Deng et al., 2021). This study adopts a one-stage approach, given its balance between speed and accuracy, which makes it suitable for practical underwater applications. Additionally, data augmentation has been widely used to enhance training datasets (Conley et al., 2022; Deng et al., 2021; Zhou et al., 2017). However, models still face challenges due to variable environmental conditions, including degraded materials, diverse color ranges, and fluctuating light intensities, all of which hinder the accurate detection of plastic debris underwater.

This paper is structured as follows. Section 2 reviews the state of the art, covering datasets in underwater environments, underwater object detection algorithms, advanced YOLO-based detection methods, and vision transformer (ViT) object detection techniques. Section 3 describes the methodology, detailing the dataset, preprocessing, deployment approach, experimental scenarios, and assessment criteria. Section 4 presents the experimental results, followed by a discussion of key findings in Section 5. Conclusions are drawn in Section 6, and directions for future research are presented in Section 7.

## STATE OF THE ART

### Datasets in Underwater Environments

The quality and quantity of images in a dataset are critical for the success of deep learning models, especially in complex environments like underwater settings. However, underwater datasets remain relatively scarce and often lack variation, posing challenges for robust model development.

One notable dataset is that of Hong et al. (2020), which comprises 7212 images of marine trash captured by a submersible remotely operated vehicle (ROV) in the Sea of Japan. It includes segmentation annotations and two versions based on object class configurations, offering flexibility for different research needs. Another is AquaTrash, developed by Panwar et al. (2020), which is based on the Trash Annotations in Context (TACO) dataset. From TACO's 1500 urban litter images, 369 were reclassified into four categories: glass, metal, paper, and plastic. While highly detailed, its urban context and limited size restrict its applicability in underwater settings. The Trash-ICRA19 dataset is widely used in underwater trash detection. It consists of 5700 images from the Japanese Environmental Data Initiative (J-EDI) database, covering 2000 to 2017 and classified into three categories. Its variability in image quality, depth, and camera types presents challenges for consistent training. The CleanSea dataset, created by Japan Agency for Marine-Earth Science and Technology (JAMSTEC), contains 1223 images categorized into 19 waste types. Its inclusion of flora and fauna adds complexity, as models must differentiate between natural and man-made objects. Moorton et al. (2022) contributed a private dataset of 1644 high-quality images featuring medium-sized trash, fishing nets, and natural elements, useful for general object classification. Similarly, Xue, Huang, Wei et al. (2021) and Xue, Huang, Chen et al. (2021) developed datasets containing 10 000 and 13 914 images, respectively, covering categories such as plastic, metal, rubber, nets, and glass. Although broad in scope, the environmental contexts of these datasets may not be exclusively underwater, limiting their specific applicability. Dhana Lakshmi and Santhanam (2020) compiled 11 797 manually labeled images from the Indian Ocean, captured at depths of 5 to 15 meters, offering targeted insights into marine waste at shallow depths. However, its scope might not represent more diverse underwater environments. Meanwhile, Zhou et al. (2017) adapted ImageNet—a large dataset with over 14 million images and 22 000 classes—for underwater recognition by selecting relevant categories. Despite its size, its general-purpose nature limits its direct application to underwater debris detection.

In summary, while several valuable datasets exist for underwater object detection, many are constrained by size, variability, or context. Table 1 presents a summary of the key characteristics of these datasets.

**Table 1**

*Datasets in Underwater Environments*

| Dataset | Images | Setting | Focus | Notes / Limitations |
|---|---|---|---|---|
| Hong et al. (2020) | 7212 | Underwater | Marine trash | Flexible object class configurations |
| AquaTrash | 369 | Urban | Glass, metal, paper, plastic | Urban context; limited underwater applicability |
| Trash-ICRA19 | 5700 | Underwater | Three categories | Variable image quality, depth, and camera types |
| CleanSea | 1223 | Underwater | 19 waste types, including flora and fauna | Presence of flora and fauna increases classification complexity |
| Moorton et al. (2022) | 1644 | Underwater | Medium-sized trash, fishing nets, natural elements | Private dataset; medium-quality images |
| Xue, Huang, Wei et al. (2021); Xue, Huang, Chen et al. (2021) | 10 000 & 13 914 | Mixed | Plastic, metal, rubber, nets, glass | Not exclusively underwater; limits specificity |
| Dhana Lakshmi & Santhanam (2020) | 11 797 | Underwater | Marine waste | Focus on shallow depths; limited diversity |
| Zhou et al. (2017) | 14+ million | General (ImageNet) | General categories | Extensive but general-purpose; limited underwater focus |

*Note.* Setting = Image capture location

### Underwater Object Detection Algorithm

Underwater object detection using deep learning has made significant progress, despite challenges such as low image quality and limited datasets. Researchers have explored various convolutional neural networks (CNNs) and object detection frameworks to improve accuracy and efficiency. Panwar et al. (2020) applied RetinaNet with a ResNet-50 backbone and feature pyramid networks (FPNs), using transfer learning to enhance performance even with smaller datasets. Similarly, Kavitha et al. (2022) achieved 98.2 % accuracy in trash detection using a lightweight three-layer CNN, aided by data augmentation and a self-curated dataset.

Other studies have focused on performance under harsh visual conditions. Rizos and Kalogeraki (2021) compared basic CNNs and ResNet50v2 in low-light settings, highlighting gradient fading. Wu et al. (2020) employed YOLOv4 for real-time detection

on ROVs, optimizing speed through lowering video resolution and enhancing image clarity using transfer learning and relative global histogram stretching (RGHS). Teng et al. (2022) introduced YOLOv5 with "predict boxes" and generalized intersection over union (GIoU) loss, improving accuracy on the Trash-ICRA19 dataset. Muksit et al. (2022) proposed YOLOFish-1 and YOLOFish-2, which enhanced feature extraction with upsampler heads and spatial pyramid pooling (SPP).

Additionally, Han et al. (2020) improved feature preservation by using CNNs with residual blocks, although training stability was an issue. Comparative studies by Fulton et al. (2019) and Conley et al. (2022) showed that Faster R-CNN and Mask R-CNN excel in accuracy, while YOLO variants offer faster inference, emphasizing the importance of selecting models based on application-specific trade-offs between speed and precision.

### Advanced YOLO Object Detection Algorithms

The YOLO architecture has undergone continuous refinement, leading to notable improvements in detection accuracy, speed, and efficiency. These advancements have established YOLO as a leading framework for real-time object detection across a wide range of domains.

Reis et al. (2023) applied YOLOv8 to aerial image detection, focusing on aircraft recognition. They compared different model sizes, finding that the small model offered a 0.05-second speed advantage over the medium one, while the large model provided only a slight improvement (0.002 seconds). Inference times were 4.1 ms (small), 5.7 ms (medium), and 9.3 ms (large). After tuning, YOLOv8 achieved an mAP50-95 of 0.835, outperforming previous benchmarks. Li et al. (2023) enhanced YOLOv8 for drone-based aerial detection by modifying the loss function, backbone, and neck, resulting in an mAP of 91.7 %. However, the model still faced challenges in detecting small objects, such as bicycles. Casas et al. (2023) compared YOLOv5 and YOLO-NAS using the Foggia dataset, a dataset designed for smoke and wildfire detection. They observed that model size correlated with training time—YOLO-NAS small, medium, and large required 2427, 3460, and 4375 hours, respectively. All models maintained high recall (0.96), which is critical for minimizing false negatives in fire detection. Terven and Córdova-Esparza (2023) highlighted YOLO-NAS's integration of automated neural architecture construction (AutoNAC), a neural architecture search (NAS) technique that automates architectural optimization. YOLO-NAS also supports 8-bit signed integer (INT8) quantization for efficient real-time inference. The study emphasized the evolution of feature extraction techniques, from early max pooling to advanced CNN backbones that combine multiscale features—such as edges and shapes—into richer feature maps. These developments significantly enhance YOLO's adaptability and accuracy in complex, real-world detection scenarios.

### ViT Object Detection Algorithms

ViT introduces a new paradigm in computer vision by applying the transformer architecture—originally developed for natural language processing—to image recognition. Like YOLO, ViT functions as a one-stage detector, making it suitable for real-time applications. In contrast to CNNs, which rely on local receptive fields, ViT uses self-attention to capture global image context, enabling it to learn complex inter-region dependencies.

Uparkar et al. (2023) compared ViTs and CNNs for lung disease classification using X-ray images. When trained from scratch, CNNs outperformed ViTs due to their strong inductive biases and better performance with limited data. However, with pre-training, ViTs slightly surpassed CNNs, achieving 1 % higher accuracy—highlighting their potential when sufficient data or optimization is available. In another study, Zhao et al. (2023) benchmarked YOLOv6, YOLOv7, YOLOv8, and a custom transformer-based detector on the Common Objects in Context (COCO) dataset. Their transformer model introduced two key architectural innovations. First, the encoder included a hybrid attention system with two modules: adaptive interaction-focused integration (AIFI), which efficiently focused attention on meaningful image regions, and cross-channel feature mixing (CCFM), which preserved fine-grained features from shallow layers. Second, a constrained optimization technique was applied to improve query selection, refining the model's attention alignment with actual object locations. These enhancements led to a 2 % improvement in AP50 and a 50 % increase in processing speed (fps) over YOLOv8, demonstrating the transformer model's superiority in both accuracy and efficiency for real-time object detection.

Table 2 groups the key algorithms discussed, offering a comparative overview to support understanding of their effectiveness and design differences.

**Table 2**

*Key Object Detection Algorithms*

| Author(s) | Model | Key Features | Dataset | Performance / Results |
|---|---|---|---|---|
| Reis et al. (2023) | YOLOv8 | Speed/accuracy tradeoff, hyperparameter tuning | Aircraft | mAP50-95: 0.835, small model fastest |
| Li et al. (2023) | Modified YOLOv8 | Custom loss, backbone, and neck for drone vision | Drone dataset | mAP: 91.7 %, improved skyborne object detection |
| Casas et al. (2023) | YOLO-NAS | AutoNAC architecture search, INT8 quantization | FOGGIA | High recall (0.96) across all sizes |
| Teng et al. (2022) | YOLOv5 + GIoU | Predict boxes, IoU-based loss penalization | Trash-ICRA19 | Improved object localization |

(*continúa*)

(*continuación*)

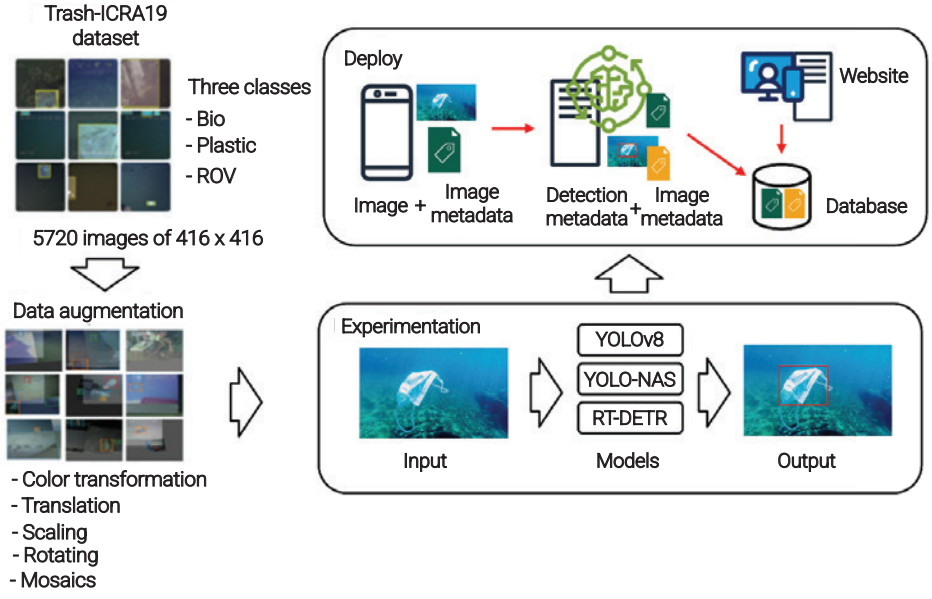
| Author(s) | Model | Key Features | Dataset | Performance / Results |
|---|---|---|---|---|
| Muksit et al. (2022) | YOLOFish-1 & YOLOFish-2 | Upsampler head, SPP module with Darknet-53 | Trash-ICRA19 | Enhanced feature extraction |
| Zhao et al. (2023) | Transform-er-based model | AIFI and CCFM modules, optimized query selection | COCO | +2 % AP50 over YOLOv8, 50 % fps boost |
| Uparkar et al. (2023) | ViT vs CNN | Self-attention, pretraining required for ViT performance | Lung X-rays | ViT outperformed CNN by 1 % with pretraining |

## METHODOLOGY

This research followed a structured implementation pipeline for underwater object detection, summarized in Figure 1, which outlines the general phases of the detection workflow. These steps include dataset selection, preprocessing, model selection and training, evaluation, and final deployment. The study focuses on analyzing and comparing deep learning-based models using metrics such as mean average precision (mAP), sensitivity, and execution time.

**Figure 1**

*General Implementation Phases and Workflow*

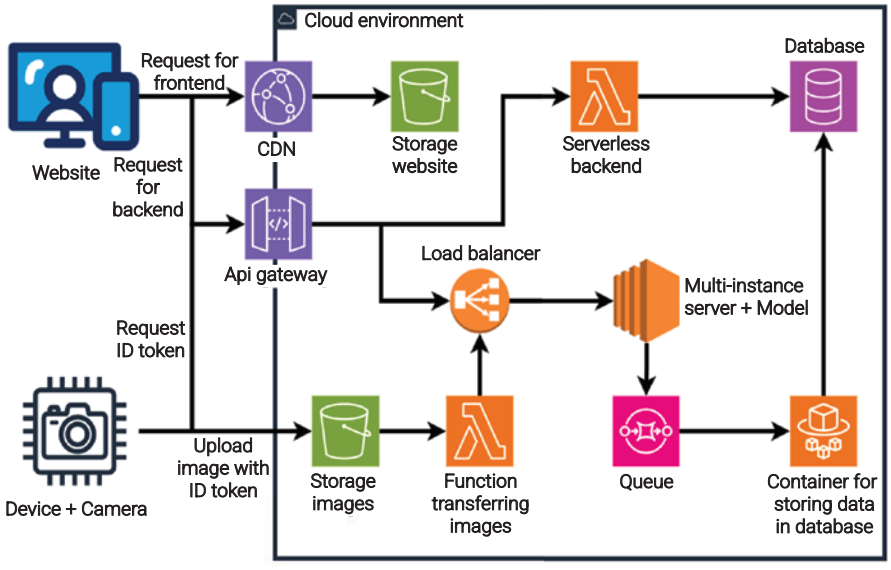


### System Overview

To illustrate the practical application of the methodological workflow outlined in Figure 1, Figure 2 depicts the system architecture designed to support scalable and efficient deployment. The design is organized into two main flows: a backend pipeline for server-side inference and data storage, and a frontend layer for presenting results to users via a web interface. A more detailed explanation of this architecture and its operational components is provided in Section 3.6.

**Figure 2**

*Cloud-Based Architecture for Deploying the Object Detection Model*




### Dataset

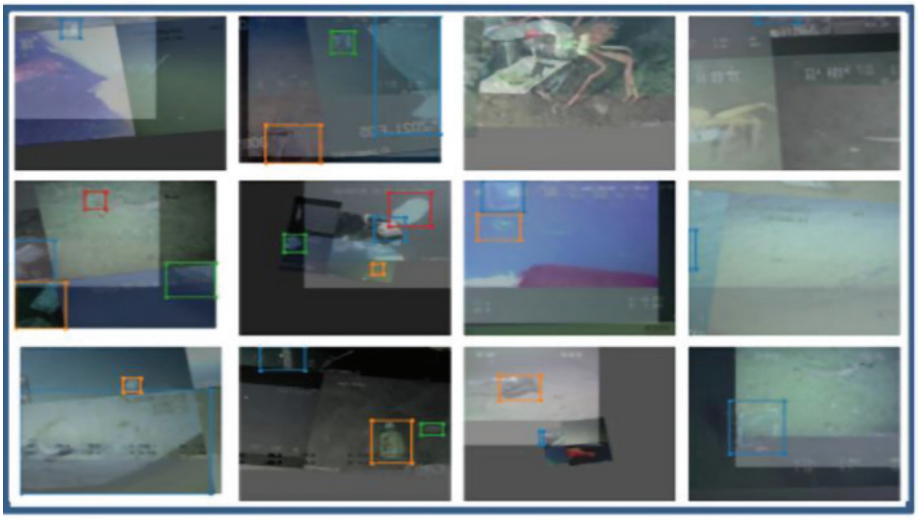
For this study, the J-EDI Trash-ICRA19 dataset was selected. It comprises 5720 underwater images (416 × 416 pixels) featuring a wide range of objects, including marine debris, animals, and equipment. The images are unprocessed and contain multiple objects of varying complexity. The dataset is categorized into three classes: plastic, remotely operated vehicle (ROV) parts, and bio (biological material). This dataset was chosen following a comparative review of available underwater datasets (see Section 2.1), where Trash-ICRA19 stood out for its balanced combination of sample size, class diversity, and high-quality annotations. In contrast to alternatives such as AquaTrash—which is focused on urban environments—or CleanSea, which offers fewer samples, Trash-ICRA19 provides greater contextual relevance for underwater debris detection. Additionally, its frequent use in previous research—particularly with models such as YOLOv5 and YOLOFish—supports its compatibility with the models employed in this study.

### Preprocessing

For training purposes, the dataset was modified in three main aspects: class distribution, image dimensions, and visual characteristics. First, the "Trash" class was prioritized to improve detection metrics, with a focus on identifying objects outside the sea. Second, all images were resized to 460 × 460 pixels to ensure compatibility with the algorithm. Finally, data augmentation techniques were applied—introducing variations in noise, mosaic patterns, exposure, and saturation—resulting in a total of 13 000 training images, as shown in Figure 3.

**Figure 3**

*Training Images with Data Augmentation*



### Experimental Scenarios

Three experimental scenarios were designed to evaluate model performance using mAP, recall, and average inference time. Scenario 1 tests YOLOv8, which is optimized for real-time accuracy in low-resource environments. Scenario 2 uses YOLO-NAS, which applies neural architecture search to balance speed and precision. Scenario 3 evaluates RT-DETR, a transformer-based model leveraging attention mechanisms for complex scene understanding. These models span a range of modern detection strategies—convolutional, hybrid, and transformer-based—enabling a comprehensive and comparative analysis across various deployment contexts.

Training was performed on Google Colab in a graphics processing unit (GPU)-enabled environment, with datasets accessed directly via Google Drive. PyTorch was used for implementation and library management. After training, models were evaluated

on a local PC equipped with an NVIDIA GeForce GTX 1660 Ti GPU, utilizing cuDNN to optimize performance. Learning rate scheduling was controlled using initial and final rates along with the total number of epochs, allowing dynamic adjustment throughout the training process.

### Deployment

The deployment follows the cloud-native architecture illustrated in Figure 3, designed to support scalable, efficient, and low-maintenance operation of the object detection system.

The first flow begins at the edge, where a camera-equipped device captures underwater images and sends them—along with an authentication token—to the cloud. These images are uploaded to cloud storage. Once stored, another serverless function is triggered to transfer and process the images. A load balancer distributes this workload to a pool of multi-instance servers, each hosting the trained detection model. These instances analyze the images in parallel and send the results to a message queue. The final outputs are handled by a storage service that writes the results into a structured database, including metadata and inference information.

The second flow handles the user-facing interface. Users interact with a web application hosted in the cloud. Requests are routed through a content delivery network (CDN) and an application programming interface (API) gateway for backend requests. The backend is managed by serverless functions. This design ensures low-latency access to results, even in low-bandwidth or edge environments. This separation of concerns ensures low latency for users by centralizing all computationally intensive tasks on the server.

### Assessment

This study evaluates object detection models—YOLO variants and RT-DETR—using three core metrics: mAP, recall, and average inference time. These metrics were selected to provide a balanced analysis of model accuracy, sensitivity, and practical deployment speed, particularly in the context of automatic waste detection.

Recall was chosen to assess the model's sensitivity, i.e., its ability to correctly detect all relevant objects. This is especially critical in waste detection tasks, where missing objects can negatively affect classification and sorting. Recall is defined mathematically as:

$$Recall = TP / (TP + FN) \quad (1)$$

Where:

TP = True positives (correctly detected objects)

FN = False negatives (missed objects)

mAP provides an overall measure of detection accuracy by combining both precision and recall across different IoU thresholds. The mAP formula is:

$$mAP = \sum_{q=1}^{Q} AveP\,(q) \,/\, Q \qquad (2)$$

Where:

Q = Number of object queries

AveP(q) = Average precision for query q

Average inference time measures how long the model takes to process an image, which is essential for evaluating real-time performance in operational environments. This metric helps determine whether a model is suitable for time-sensitive applications.

Together, these three metrics offer a concise and representative framework to evaluate model performance in terms of detection accuracy, sensitivity, and processing efficiency.

## RESULTS

Initial experimentation highlighted the most salient comparative characteristics of the evaluated models.

**Table 3**

*Scenarios for Model Comparison*

| Models | mAP50 | mAP50-95 | Re | IT | TT |
|---|---|---|---|---|---|
| YOLOv8 S | 0.821 | 0.534 | 0.720 | 10.1* | 3 |
| YOLOv8 M | 0.833 | 0.589 | 0.750 | 19.4* | 5 |
| YOLOv8 L | 0.807 | 0.564 | 0.671 | 35* | 13 |
| YOLO-NAS S | 0.735 | 0.668 | 0.967 | 2.04 | 3 |
| YOLO-NAS M | 0.736 | 0.681 | 0.856 | 3.14 | 4 |

*Note.* mAP50 = Mean average precision at an IoU threshold of 0.50; mAP50-95 = Mean average precision at IoU thresholds from 0.50 to 0.95; Re = Recall (sensitivity); IT = Inference time in milliseconds; TT = Training time in hours; inference time marked with an asterisk (*) was measured using NVIDIA GeForce GTX 1660 Ti GPU.

Table 3 summarizes the performance of the YOLOv8 and YOLO-NAS models on the Trash-ICRA19 dataset. Among YOLOv8 variants, the M variant demonstrated the best overall performance, achieving an mAP50 of 0.833, an mAP50-95 of 0.589, a recall of 0.75, and an inference time of 19.4 ms, making it the most balanced choice for further tuning. The S variant followed, offering slightly lower accuracy (mAP50: 0.821) but faster inference (10.1 ms). In comparison, the L model had lower recall (0.671) and significantly higher inference time (35 ms), indicating less efficiency despite a modest mAP50-95 of 0.564. In contrast, the YOLO-NAS models excelled in recall, with the S variant achieving 0.967 and the M variant reaching 0.856. Although their mAP50 values (0.735 and 0.736, respectively) were lower than those of YOLOv8, their mAP50-95 scores were competitive. Additionally, both YOLO-NAS models required less training times, with the S and M variants completing training in 3 and 4 hours, respectively. These results suggest that YOLO-NAS, particularly the S variant, is a practical option under limited computational resources, offering strong recall and efficient training while maintaining reasonable detection accuracy.

**Table 4**

*Hyperparameter Refinement of YOLOv8 M*

| P | F | Bat | Mo | Ep B | lr | Ep T | mAP50 | mAP50-95 | Re |
|---|---|---|---|---|---|---|---|---|---|
| - | 0 | 32 | Yes | 100 | 0.001 | 100 | 0.833 | 0.589 | 0.75 |
| P1 | 0 | 32 | No | 46 | 0.001 | 50 | 0.834 | 0.557 | 0.72 |
| P2 | 5 | 48 | Yes | 11 | 0.001 | 50 | 0.866 | 0.626 | 0.728 |
| P3 | 5 | 48 | Yes | 22 | 0.0005 | 50 | 0.888 | 0.633 | 0.808 |
| P4 | 5 | 48 | Yes | 13 | 0.0001 | 30 | 0.906 | 0.627 | 0.828 |
| P5 | 0 | 32 | Yes | 18 | 0.0001 | 30 | 0.922 | 0.644 | 0.829 |

*Note.* P = Test number; F = Number of frozen layers for transfer learning; Bat = Batch size; Mo = YOLO native data augmentation; Ep B = Best epoch; lr = Learning rate; Ep T = Total number of epochs; mAP50 = Mean average precision at an IoU threshold of 0.50; mAP50-95 = Mean average precision at IoU thresholds from 0.50 to 0.95; Re = Recall (sensitivity).

Table 4 presents the results of hyperparameter optimization for YOLOv8 M. Five tuning scenarios were tested. In P1, disabling native data augmentation resulted in performance drops (mAP50 = 0.834, mAP50-95 = 0.557, Re = 0.72), confirming the importance of YOLO's built-in augmentation. Transfer learning was explored by freezing different numbers of layers. Tests P2 through P4 revealed that freezing fewer layers and reducing the number of training epochs improved performance. In P5, further tuning of the learning rate and configuration achieved the best results: mAP50 = 0.922, mAP50-95 = 0.644, and Re = 0.829.

**Table 5**

*Hyperparameter Refinement of the YOLO-NAS S Model (30 Epochs)*

| P | Bat | Mo | Ep B | lr | mAP50 | Re |
|---|---|---|---|---|---|---|
| - | 16 | Yes | 30 | 0.001 | 0.682 | 0.860 |
| P1 | 16 | Yes | 28 | 0.001 | 0.713 | 0.916 |
| P2 | 16 | Yes | 19 | 0.001 | 0.716 | 0.883 |
| P3 | 16 | Yes | 15 | 0.0005 | 0.627 | 0.824 |
| P4 | 16 | Yes | 8 | 0.0001 | 0.694 | 0.923 |
| P5 | 16 | Yes | 4 | 0.0001 | 0.735 | 0.967 |
| P6 | 16 | Yes | 4 | 0.0001 | 0.813 | 0.903 |

*Note.* P = Test number; Bat = Batch size; Mo = YOLO native data augmentation; Ep B = Best epoch; lr = Learning rate; mAP50 = Mean average precision at an IoU threshold of 0.50; Re = Recall (sensitivity).

Table 5 presents the results of hyperparameter tuning for the S variant of YOLO-NAS. The baseline model, trained for 30 epochs, achieved an mAP50 of 0.682 and a recall of 0.860. Adjustments to the optimizer (Adam), learning rate, and early stopping—particularly in scenarios P1 to P3—led to a notable increase in recall, reaching up to 0.923, while mAP values showed moderate variation. Between epochs 8 and 15, mAP remained relatively stable, with consistently high recall values. The best overall performance was recorded in P6, where the optimizer was changed to stochastic gradient descent (SGD) and the confidence threshold was set to 0.25. This configuration resulted in an mAP50 of 0.813 and a recall of 0.903.

**Table 6**

*Performance Metrics and Characteristics of Selected Detection Models*

| Models | Parameters | Layers | mAP50 | mAP50-95 | Re | IT |
|---|---|---|---|---|---|---|
| YOLOv8 M | 25 902 640 | 295 | 0.921 | 0.644 | 0.829 | 14.1 |
| YOLOv8 L | 43 691 520 | 365 | 0.915 | 0.650 | 0.813 | 19.2 |
| YOLO-NAS S | 19 053 888 | 685 | 0.813 | - | 0.903 | 17.8 |
| RT-DETR L | 32 970 476 | 673 | 0.887 | 0.589 | 0.819 | 15.9 |

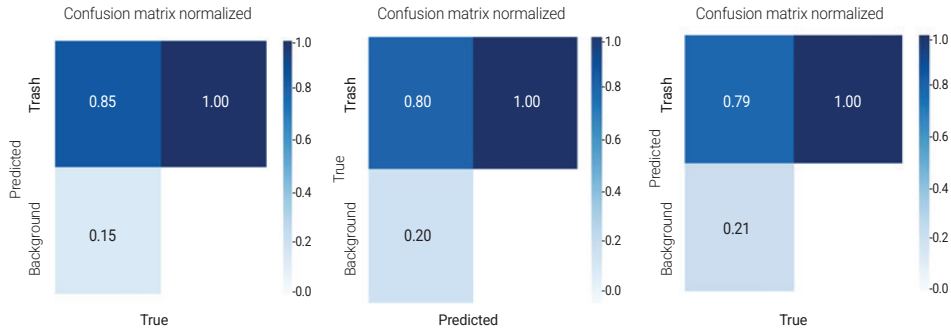
*Note.* mAP50 = Mean average precision at an IoU threshold of 0.50; mAP50-95 = Mean average precision at IoU thresholds from 0.50 to 0.95; Re = Recall (sensitivity); IT = Inference time in milliseconds.

Table 6 shows that the first scenario, where YOLOv8 M was used to classify and detect plastic waste, obtained an mAP50 of 0.921, an mAP50-95 of 0.644, a recall of 0.829, and an inference time of 14.1 milliseconds. In the second scenario, YOLO-NAS S reached an mAP50 of 0.813, a fairly high recall of 0.903, and an inference time of 17.8 milliseconds. In the third scenario, using RT-DETR L, the model achieved an mAP50 of

0.887, an mAP50-95 of 0.589, a recall of 0.819, and an average inference time of 15.9 milliseconds, slightly lower compared to YOLOv8 M.

**Figure 4**

*Standardized Confusion Matrices of the Models*




*Note.* The confusion matrix of YOLOv8 is shown on the left, YOLO-NAS in the center, and RT-DETR on the right.

Figure 4 presents the detection performance of the three evaluated models. For YOLOv8, 85 % of the total garbage instances were correctly identified as true positives, corresponding to 719 detections. The remaining 15 %, or 134 instances, were classified as false negatives, indicating that they were not detected as garbage despite belonging to the target class. Regarding the background class, YOLOv8 exhibited a 100 % false positive rate, with 81 background instances incorrectly identified as garbage. Similarly, YOLO-NAS correctly detected 80 % of garbage instances, while the remaining 20 % were false negatives, i.e., relevant objects were missing. As with YOLOv8, the background was entirely misclassified, resulting in a 100 % false positive rate and 81 incorrect detections. The RT-DETR confusion matrix showed a comparable trend. It correctly identified 79 % of garbage instances as true positives, while 21 % were false negatives. Like the other models, the background was completely misclassified, yielding a 100 % false positive rate.

**Figure 5**

*Model Inferences Across Scenarios*


*Note.* YOLOv8 M inference (top left), YOLOv8 L inference (top right), YOLO-NAS S inference (bottom left), and RT-DETR inference (bottom right).

Figure 5 illustrates the detections made by the models in a video that resembles a real-world environment. Notably, the training dataset consisted of underwater bottom trash images captured under very low illumination, unlike a video of a real-world environment with much higher illumination. Under these conditions, the RT-DETR model demonstrated superior performance in detecting small objects compared to the YOLOv8 and YOLO-NAS models.

## DISCUSSION

Data augmentation played a vital role in this study, particularly given the medium-sized dataset. It enhanced data diversity and improved the models' ability to generalize to unseen scenarios, thereby helping to mitigate overfitting. Model selection was guided by computational resources and model complexity. YOLO-NAS S was chosen for its training efficiency, while the M or L variants of other models were employed to balance performance with available resources. Traditional approaches such as dataset splitting and augmentation were prioritized due to hardware constraints. Although cross-validation can offer more robust evaluation, it was avoided due to the increased computational cost and the characteristics of the dataset.

During deployment, a key trade-off emerged between video resolution and detection speed. Higher resolutions reduced false positives by improving visual clarity but decreased processing speed (i.e., lower frames per second [FPS]). This trade-off is especially relevant in visually challenging environments, such as underwater scenes affected by sand or debris. These findings highlight the importance of image preprocessing and resolution tuning in real-world applications.

While precision was a critical evaluation metric, recall proved equally important. RT-DETR achieved the highest precision, but YOLO-NAS excelled in recall, ensuring broader object detection coverage. In applications like automatic waste detection or trash mapping, high precision is especially important: missing items such as bottles or hazardous waste can reduce sorting efficiency and pose environmental or safety risks. In such cases, detecting more objects—even at the cost of occasional false positives—is preferable. This study underscores the importance of balancing precision and recall based on task context, highlighting YOLO-NAS's strength in recall-focused scenarios.

These results are consistent with previous research. Terven and Córdova-Esparza (2023) highlighted YOLO's ability to extract key features such as edges and textures—findings that are validated in this study. Additionally, ViT models performed well under visual noise, leveraging attention mechanisms for contextual recognition. This supports findings of Maurício et al. (2023), underscoring ViT's robustness in degraded conditions and its value in challenging environments.

## CONCLUSIONS

This research successfully achieved its initial objectives. Beyond acquiring the expected knowledge, the study yielded meaningful conclusions regarding the comparative performance of the evaluated models. Training various models with the Trash-ICRA19 dataset broadened our understanding of deep learning workflows, including critical concepts such as epochs, loss functions, and evaluation metrics. A thorough analysis of each model's architecture further enhanced our comprehension of their functionality from training to real-world deployment.

Among the models tested, RT-DETR demonstrated the most promising performance in practical scenarios. Although it did not achieve the highest validation scores on the dataset, its superior real-world performance underscores the importance of evaluating models in real environments, where traditional metrics may not fully capture practical effectiveness.

To address the limitations posed by resource-constrained environments, this study proposed offloading the recognition logic to a powerful cloud or edge server. While this approach introduces potential latency due to data transmission, it allows

the use of more complex and accurate models that would otherwise be infeasible on low-power devices.

In conclusion, RT-DETR proves to be a reliable starting point for underwater waste detection, especially in identifying garbage bags. Its successful deployment in real-world scenarios—such as the beaches of Bali—demonstrates its potential for integration into ROVs. These systems could be used to detect and map plastic waste accumulation zones, thereby contributing to more efficient and targeted environmental cleanup efforts.

## FUTURE RESEARCH

Several enhancements are recommended for future research. First, increasing the resolution of input images may improve object detection accuracy. Incorporating a classification component would further enhance system capabilities to identify specific waste types, such as garbage bags, wrappers, or bottles. Additionally, using a new or more diverse dataset—ideally one tailored to underwater conditions—would support better model robustness and generalization.

Another key area for improvement is the reduction of transmission delays. This could be achieved by initially lowering image resolution to enable faster transmission and then applying a super-resolution algorithm on the server to enhance image quality before feeding it into the detection model. In this way, computational load is entirely offloaded to high-performance servers, optimizing both speed and accuracy.

## REFERENCES

Casas, E., Ramos, L., Bendek, E., & Rivas-Echeverría, F. (2023). Assessing the effectiveness of YOLO architectures for smoke and wildfire detection. *IEEE Access*, *11*, 96554–96583. https://doi.org/10.1109/access.2023.3312217

Conley, G., Zinn, S. C., Hanson, T., McDonald, K., Beck, N., & Wen, H. (2022). Using a deep learning model to quantify trash accumulation for cleaner urban stormwater. *Computers, Environment and Urban Systems*, *93*, Article 101752. https://doi.org/10.1016/j.compenvurbsys.2021.101752

Deng, H., Ergu, D., Liu, F., Ma, B., & Cai, Y. (2021). An embeddable algorithm for automatic garbage detection based on complex marine environment. *Sensors*, *21*(19), Article 6391. https://doi.org/10.3390/s21196391

Dhana Lakshmi, M., & Santhanam, S. M. (2020). Underwater image recognition detector using deep ConvNet. 2020 National Conference on Communications (NCC), Kharagpur, India, pp. 1–6. http://doi.org/10.1109/NCC48643.2020.9056058

Fulton, M., Hong, J., Islam, M. J., & Sattar, J. (2019). Robotic detection of marine litter using deep visual detection models. *2019 International Conference on Robotics and Automation (ICRA),* Montreal, QC, Canada, pp. 5752–5758. https://doi.org/10.1109/ICRA.2019.8793975

Han, F., Yao, J., Zhu, H., & Wang, C. (2020). Underwater image processing and object detection based on deep CNN method. *Journal of Sensors*, *2020*, Article 6707328, 1–20. https://doi.org/10.1155/2020/6707328

Hong, J., Fulton, M., & Sattar, J. (2020). Trashcan: A semantically-segmented dataset towards visual detection of marine debris. *arXiv*. https://doi.org/10.48550/arXiv.2007.08097

Kavitha, P. M., Anitha, M., Padhiari, S., & Anitha, K. (2022). Detection of trash in sea using deep learning. *YMER Digital*, *21*(7), 817–822. https://doi.org/10.37896/ymer21.07/65

Li, Y., Fan, Q., Huang, H., Han, Z., & Gu, Q. (2023). A modified YOLOv8 detection network for UAV aerial image recognition. *Drones*, *7*(5), Article 304. https://doi.org/10.3390/drones7050304

Maurício, J., Domingues, I., & Bernardino, J. (2023). Comparing vision transformers and convolutional neural networks for image classification: A literature review. *Applied Sciences*, *13*(9), Article 5521. https://doi.org/10.3390/app13095521

Moorton, Z., Kurt, Z., & Woo, W. L. (2022). Is the use of deep learning an appropriate means to locate debris in the ocean without harming aquatic wildlife? *Marine Pollution Bulletin*, *118*, Article 113853. https://doi.org/10.1016/j.marpolbul.2022.113853

Muksit, A. A., Hasan, F., Hasan Bhuiyan Emon, M. F., Haque, M. R., Anwary, A. R., & Shatabda, S. (2022). YOLO-Fish: A robust fish detection model to detect fish in realistic underwater environment. *Ecological Informatics*, *72*, Article 101847. https://doi.org/10.1016/j.ecoinf.2022.101847

National Institute of Statistics and Informatics. (2022). *Technical report: Environmental statistics for January 2022*. https://cdn.www.gob.pe/uploads/document/file/3067334/Estad%C3%ADsticas%20Ambientales%3A%20Enero%202022.pdf?v=1651873783

Panwar, H., Gupta, P. K., Siddiqui, M. K., Morales-Menendez, R., Bhardwaj, P., Sharma, S., & Sarker, I. H. (2020). AquaVision: Automating the detection of waste in water bodies using deep transfer learning. *Case Studies in Chemical and Environmental Engineering*, *2*, Article 100026. https://doi.org/10.1016/j.cscee.2020.100026

Reis, D., Kupec, J., Hong, J., & Daoudi, A. (2023). *Real-time flying object detection with YOLOv8*. arXiv. https://doi.org/10.48550/arXiv.2305.09972

Rizos, P., & Kalogeraki, V. (2021). Deep learning for underwater object detection. *Proceedings of the 24th Pan-Hellenic Conference on Informatics (PCI '20)*, New York, USA, pp. 175–177. https://doi.org/10.1145/3437120.3437301

Teng, X., Fei, Y., He, K., & Lu, L. (2022). The object detection of underwater garbage with an improved YOLOv5 algorithm. Proceedings of the 2022 International Conference on Pattern Recognition and Intelligent Systems (PRIS '22), New York, USA, pp. 55–60. https://doi.org/10.1145/3549179.3549189

Terven, J., & Córdova-Esparza, D. M. (2023). A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLONAS. *Machine Learning and Knowledge Extraction*, 5(4), 1680–1716. https://doi.org/10.3390/make5040083

Uparkar, O., Bharti, J., Pateriya, R. K., Gupta, R. K., & Sharma, A. (2023). Vision transformer outperforms deep convolutional neural network-based model in classifying X-ray images. *Procedia Computer Science*, *218*, 2338–2349. https://doi.org/10.1016/j.procs.2023.01.209

Wu, Y.-C., Shih, P.-Y., Chen, L.-P., Wang, C-.C., & Samani, H. (2020). Towards underwater sustainability using ROV equipped with deep learning system. *2020 International Automatic Control Conference (CACS)*, Hsinchu, Taiwan, pp. 1–5. http://doi.org/10.1109/CACS50047.2020.9289788

Xue, B., Huang, B., Wei, W., Chen, G., Li, H., Zhao, N., & Zhan, H. (2021a). An efficient deep-sea debris detection method using deep neural networks. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 14, 12348–12360. http://doi.org/10.1109/JSTARS.2021.3130238

Xue, B., Huang, B., Chen, G., Li, H., & Wei, W. (2021b). Deep-sea debris identification using deep convolutional neural networks. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 14, 8909–8921 http://doi.org/10.1109/JSTARS.2021.3107853

Zhao, Y., Lv, W., Xu, S., Wei, J., Wang, G., Dang, Q., Liu, Y., & Chen, J. (2023). DETRs beat YOLOs on real-time object detection. *arXiv*, 2304.08069. https://doi.org/10.48550/arXiv.2304.08069

Zhou, H., Huang, H., Yang, X., Zhang, L., & Qi, L. (2017). Faster R-CNN for marine organism detection and recognition using data augmentation. Proceedings of the International Conference on Video and Image Processing - ICVIP 2017. https://doi.org/10.1145/3177404.3177433