Journal of Real-Time Image Processing (2025) 22:108
https://doi.org/10.1007/s11554-025-01682-2

RESEARCH

Yolo-based power-efficient object detection on edge devices for USVs

Jose Luis Mela[1,2] · **Carlos García Sánchez**[1]

Received: 28 February 2025 / Accepted: 10 April 2025 / Published online: 6 May 2025

Abstract
Advances in Artificial Intelligence, the Internet of Things, and Computer Vision have introduced challenges in minimizing resource usage-economically, energetically, and environmentally. This work presents a vision system for unmanned surface vehicles (USVs) focused on object detection and autonomous navigation. The system leverages hardware and software acceleration to enhance model performance while also evaluating energy efficiency. In this paper, we analyze the Ultralytics models across various platforms, including MYRIAD VPU, Intel CPUs/GPUs, and NVIDIA Jetson AGX Orin and Orin Nano. The results show that the Orin Nano is especially suitable for real-time detection, consuming less than 2 watts. To increase efficiency, optimization techniques, such as quantization and pruning, are performed. We also compare our models with related studies and assess YOLOv6 to YOLO11 in terms of inference time and FPS. YOLOv8-based models consistently deliver the best results, confirming their suitability for USV applications.

Keywords Neural networks · Object detection · YOLO · OpenVINO · Power efficiency · Jetson Orin

1 Introduction

Advances in Artificial Intelligence (AI) have enabled the development of models for complex tasks with greater accuracy, supported by improved computing capabilities from Continuum Cloud to Edge Computing [1]. IoT technologies have introduced edge computing, which processes data close to its source, ensuring real-time functionality, reducing latency [2], optimizing energy consumption [3], and improving privacy [4]. This approach facilitates IoT data processing [5], minimizing reliance on centralized systems, which contributes to creating smarter and more reliable solutions that can be leveraged in artificial intelligence models.

Although public datasets, such as ImageNet, MS COCO, and KITTI [6–8], have driven advances in computer vision, maritime environments present unique challenges, including sunlight reflections and fog, which require optimized models for greater accuracy and efficiency.

Machine Learning (ML), a foundational component of AI, is pivotal across various fields. In particular, ML-based computer vision is critical in marine exploration and surveillance [9]. Uncrewed surface vessel systems especially require efficient object detection with low latency and minimal energy consumption.

This work is part of a larger project that aims to monitor and detect harmful algae blooms using digital twins to improve water quality [10]. Data are collected by different sensors deployed on platforms such as aquatic drones or USVs [11], which collect water samples to detect harmful agents such as cyanobacteria and their toxic status. In this context, the need arises to focus specifically on the design and acceleration of a vision system for USVs, addressing the challenge of creating high-performance and low-power consumption neural network models based on object detection in aquatic environments and applying some optimization techniques.

The main contributions of this work are threefold, with a strong focus on novelty and practical impact: (1) the design and deployment of a novel, power-efficient vision system optimized for real-time object detection on Uncrewed Surface Vessels (USVs) operating in challenging aquatic environments; (2) a thorough, hardware-aware benchmarking

✉ Carlos García Sánchez
garsanca@ucm.es

Jose Luis Mela
jmela@ucm.es

[1] Fac. Informática, Universidad Complutense de Madrid, 28040 Madrid, Spain

[2] Fac. Informática, Electrónica y Comunicación, Universidad de Panamá, Santiago de Veraguas, Panama

of recent YOLO object detection models across diverse edge platforms, including the Intel CPUs/NPUs/GPUs, and NVIDIA Jetson AGX Orin and Orin Nano, offering a unique insight into the trade-offs between accuracy, speed, and power consumption; and (3) the first integration of such a system into a digital twin framework for the monitoring and early detection of harmful algae blooms, introducing a novel AI-powered application in fresh water monitoring.

The manuscript is structured as follows: Sect. 2 reviews the state-of-the-art; Sect. 3 describes the methodology and experiments; Sect. 4 presents the results and optimizations; Sect. 5 discusses the findings and the final prototype; and Sect. 6 provides the conclusions and final remarks.

2 Background

2.1 Deep learning algorithms

Deep learning, a subset of ML, uses algorithms inspired by the human brain (neurons) to recognize patterns in data [12, 13]. Training these algorithms requires particular labeled datasets, depending on the context of use. CNNs work fine to recognize patterns in images. They consist of convolutional, pooling, and fully connected layers [14]. These layers form a structured sequence, starting with input layers, followed by hidden layers with activation functions (e.g., ReLU), and ending with fully connected layers for classification.

2.2 Deep learning frameworks

Several frameworks facilitate deep learning development [15]. In this study, we used the following:

- YOLO (You Only Look Once): An efficient object detection framework based on PyTorch developed by Ultralytics. It also excels in segmentation, classification, and tracking in a single pass [16].
- TensorFlow: A versatile framework supporting CPU, GPU, and TPU, leveraging computational graphs for efficient data processing.
- PyTorch: Optimized for deep learning tasks, focusing on tensors for data manipulation with high speed and numerical efficiency [17].
- OpenVINO: A toolkit for optimizing and deploying deep learning models on Intel devices, including CPUs, GPUs, and neural accelerators [18].

2.3 Related work

Object detection has been widely studied in various fields [19], particularly in the context of artificial intelligence and computer vision. Most research generally focuses on optimizing models to meet real-time accuracy requirements using accelerators such as GPUs, with limited emphasis on energy efficiency, especially in aquatic environments.

A recent study by Azevedo et al. [20] compares several state-of-the-art object detectors, including YOLOv8, YOLOv7, YOLOv5, Scaled-YOLOv4, and YOLOR, trained on the BDD100 K dataset, and examines their use on edge devices such as the NVIDIA Jetson AGX Xavier. Their research shows improvements in object detection using these models, but focuses on land-based vehicles without considering energy consumption or applications for unmanned surface vehicles. Similarly, Lema et al. [21] evaluated object detection algorithms, such as YOLOv3 and YOLOv5, on NVIDIA Jetson Nano, NVIDIA Jetson AGX Xavier, and Google Coral Dev Board devices, using MS COCO dataset. This work focuses on analyzing the relationship between FPS/energy consumption and FPS/cost, providing a useful framework for real-time vision systems, but does not delve into energy performance in the context of USVs.

An interesting study by Fabrizio et al. [22] examined power consumption on Google Coral and NVIDIA Jetson Nano devices, finding that power usage could vary by up to three times depending on the algorithm and hardware configuration. However, this work primarily focuses on general high-performance vision systems and does not address object detection in aquatic environments.

Several studies have explored the use of object detectors such as YOLO in the context of unmanned surface vehicles and binocular image fusion [23, 24]. For example, Liu et al. [25] present advancements in intelligent railway systems, while Dong et al. [26] introduce an optimized YOLOv7 variant featuring a multiscale pyramidal network structure and an improved loss function. However, their study does not focus on performance and energy consumption in edge devices. This suggests that, although the use of YOLO in vision for object detection has led to significant improvements in accuracy and inference times [27], its applications have focused primarily on road traffic [28–30] or agricultural detection [31–33]. In addition, there are other studies, such as those by [34–36], which use binocular ZED cameras for different tasks. However, there is a notable lack of studies on machine vision with accelerators that also consider the trade-off between power consumption and accuracy in the latest versions of YOLO, as well as acceleration strategies to reduce power usage in the context of aquatic drones with ZED cameras.

In addition, a recent study by Trinh et al. [37] provides a comprehensive review of datasets and deep learning techniques applied to vision in USVs, highlighting existing gaps in model optimization and energy efficiency in aquatic environments. This work emphasizes the specific challenges of maritime vision tasks and reinforces the need for further

research on the integration of efficient neural network models into real-world USV applications.

A key factor in energy efficiency and accuracy is model optimization, where techniques such as quantization and pruning are used to reduce the size of the model without significantly sacrificing accuracy. In this regard, Ding et al. [38] highlight that these systems have special requirements in terms of real-time implementation and energy efficiency, emphasizing the importance of model compression.

One criterion to consider is the impact of the aquatic environment, which affects both accuracy and energy efficiency in USV systems, as environmental variations can require more computational processing. Zhang et al. [39] studied environmental factors such as turbulent waves, water reflections, and fog, achieving good results with their proposal. However, they noted that the architecture could be further optimized for better performance.

Various solutions employ different neural network model architectures to balance accuracy and inference time. However, studies evaluating model optimization in conjunction with performance and energy efficiency are scarce. In this context, our work addresses this gap by applying hardware and software optimization techniques to enhance both performance and energy efficiency on devices such as the Intel UP Squared Pro 7000 Edge, the NVIDIA Jetson AGX Orin, and the Orin Nano, with a specific focus on unmanned surface vehicles.

3 Methodology and experiments

Figure 1 illustrates the workflow followed in this study.

The methodology is divided into four distinct phases, each involving various technologies and tools. The *preprocessing phase* consists of standardizing each dataset to the YOLOv8 format, on the Roboflow,[1] where the compilation, splitting, and application of some filters to the images were carried out. The *training phase* focuses on the neural network training process in each of the datasets. This training was run on a high-performance system equipped with high-end GPUs. Additionally, the ClearML[2] platform was used to monitor the training process of each neural network model, which allows us to graphically visualize the behavior of the model according to performance metrics. The *testing phase* involves evaluating various edge computing systems. Specifically, two leading market manufacturers, Intel and NVIDIA, have been selected for their systems equipped with low-power GPUs to run neural networks. Finally, the *deploy phase* focuses on the integration of a binocular camera with the drone to combine AI models for object detection with computing capabilities for object distance.

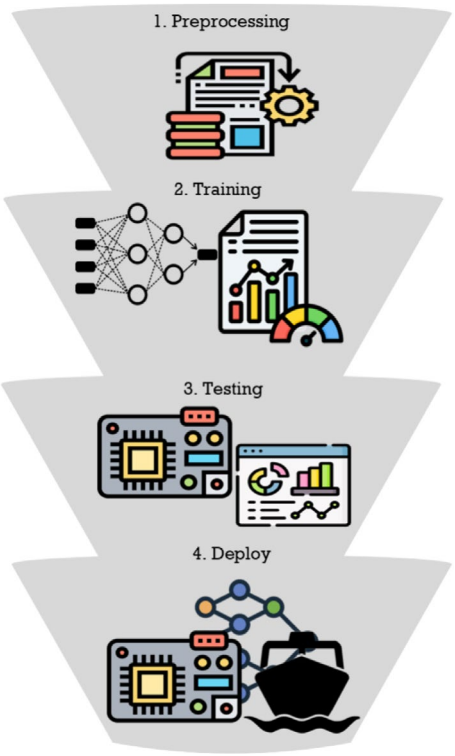

Fig. 1 Workflow

3.1 Dataset description and preprocessing

Three datasets were chosen for this study, the main characteristics of which are presented in Table 1. These datasets have been used in other research [40, 41]; therefore, it is important to note that although they contain similar types of images with equivalent objects, viewing angles vary, from zenith to frontal views. In addition, the images in these datasets consider various conditions encountered in the marine environment, such as light reflections, waves, and other factors.

The three datasets are described below:

– SeaDronesSee v2: This dataset contains zenith view images, which have five different classes of objects (see Table 1). For our case, Roboflow: Preprocess SeaDronesSee dataset, where when loading the images, it was divided into three groups (train, valid, and test—Fig. 2) and data augmentation techniques (auto-orient, resize, and grayscale) were applied. Therefore, the use of Roboflow was essential to carry out this task to work with the dataset in YOLOv8 format.

[1] Roboflow web-page: https://roboflow.com.

[2] ClearML web-page: https://clear.ml.

Table 1 Datasets used

Dataset	Vision angle	Images	Annotation	Categories
SeaDronesSee v2 [40]	Aerial	9984	61,974	Boat, Buoy, Jetski, Life saving appliances, Swimmer
Datasense@CRAS [41]	Onboard	9021	21,135	Bulk carrier, Container ship, Cruise ship, Ferry boat, Ore carrier, Sail boat, Uncategorized
USVDD	Onboard	8520	19,852	Buoy, Boat, Gondola, Jet-ski, Surfboard, Watercraft

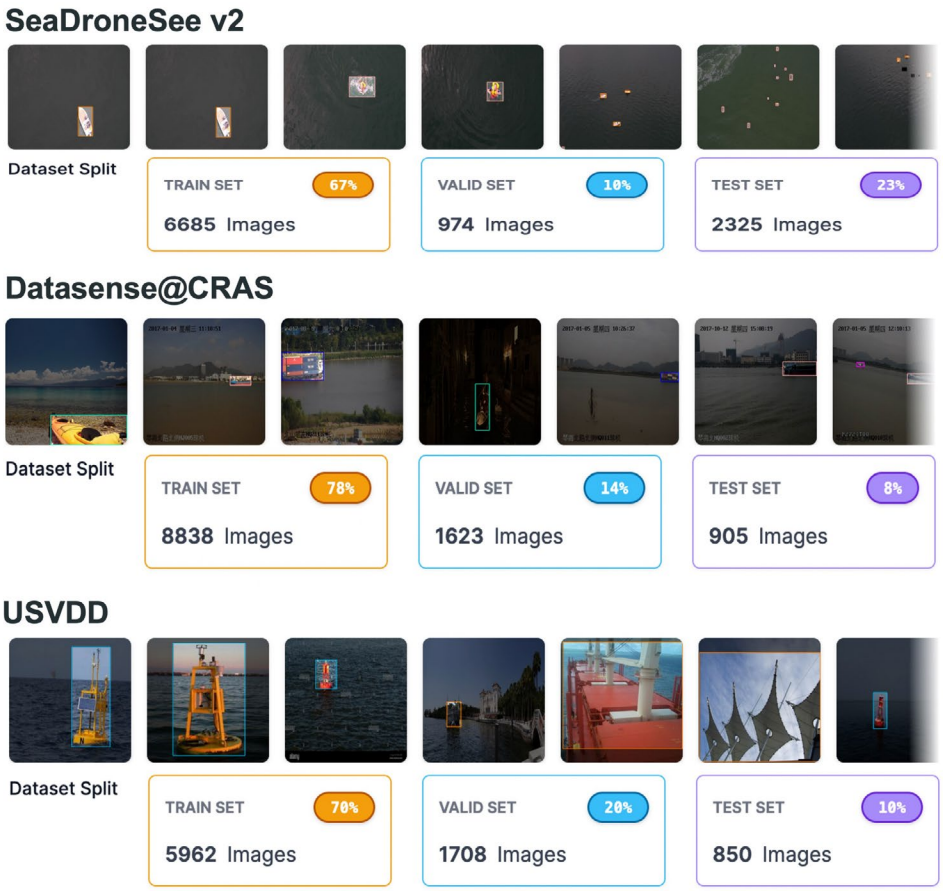

Fig. 2 Split of each dataset in the roboflow platform

– Datasense@CRAS: This dataset, as indicated by its authors, is a compilation of images from the Singapore Maritime Dataset and the Kaggle Boat Types Recognition. To work with this dataset, processing was performed in Roboflow: Preprocess Datasense@CRAS dataset, applying the same technique as the previous one but also modifying the classes, to put the names of the classes and not a number, as originally.
– USVDD: This dataset is one of our contributions and consists of classes of objects similar to the previous ones, so it may be unified in future work. These images and annotations were downloaded from OpenImages V6. To work with the dataset in the YOLOv8 format, the processing of the compiled images and annotations was done in Roboflow: Preprocess the OpenImages V6 collection with respect to aquatic images, applying the same filters as the SeaDronesSee v2 dataset and also splitting the images into three groups (see Fig. 2).

3.2 Experimental setup

The equipment used for experimentation, specifically for the training process, has the specifications given in *part a* of Table 2. Likewise, the experimental parameters for the model training phase are set, as shown in *part b* of Table 2.

Table 2 Experimental setup and training parameters for model evaluation

a) Experimental setup		
Component	Specification	
Processors	2x Intel Gold 6138	
Memory	96 GB DDR4	
GPUs	2x Tesla V100 (32 GB each)	
	1x RTX 3090 (24 GB)	
Operating System	Linux¬6.1.0–18-amd64	
Programming Language	Python 3.11	
Deep Learning Framework	PyTorch≥1.8	
b) Experimental parameters setting		
Dataset	Argument	Parameters
SeaDroneSee v2, Datasense@CRAS, USVDD	Learning rate	0.01
	Momentum	0.937
	Weight decay	0.0005
	Batch size	16
	Image size	640x640
	Epochs	150

3.3 Training model selection

The CNN used in this project is the YOLO Ultralytics family, specifically the nano, small, and medium versions of YOLOv6, YOLOv8, YOLOv9, YOLO10, and YOLO11. These versions of neural networks are chosen mainly because they have high performance in object detection tasks, standing out in recent years for their accuracy, speed, and ease of use, which makes them suitable for our project focused on image-based object detection.

3.4 Edge computing devices

The evaluation of the model previously trained has been performed on two NVIDIA-based edge computing boards equipped with an Ampere GPU denoted NVIDIA Jetson Orin (AGX and Nano versions) and on an Intel edge device (UP Squared Pro 7000 Edge + Myriad). The main features and configurations are detailed in Table 3. The AGX board, designed for industrial environments with high computational demands, offers four times the performance of its

Table 3 Technical specifications of the Jetson Orin Nano, Jetson AGX Orin, and UP Squared Pro 7000 Edge

		Jetson Orin Nano	Jetson AGX Orin	UP Squared Pro 7000 Edge
CPU	Name	ARM Cortex-A78 AE	ARM Cortex-A78 AE	Intel Atom X7425E
	Frequency	up to 1.5 GHz	up to 2.2 GHz	up to 3.40 GHz
	Cores	6	12	4
GPU	Name	Ampere GPU	Ampere GPU	UHD Graphic Gen 12 th
	VPU			Hailo-8^{TM}M.2 2280
	Frequency	625 MHz	1.3 GHz	1 GHz
	Cores	1,024 CUDA cores	2,048 CUDA cores	24 execution units
	Perf. (fp32)	1,280 GFLOPS	5,325 GFLOPS	460.8 GFLOPS
Memory		8 GB	64 GB	8 GB
Power consumption		7 W/15 W	4 modes	12 W
Price		$499	$1,999	$399

The power modes for the Jetson AGX Orin are: 15 W, 30 W, 50 W, and MAXN. Meanwhile, for the Jetson Orin Nano, the power modes are 7 W and 15 W

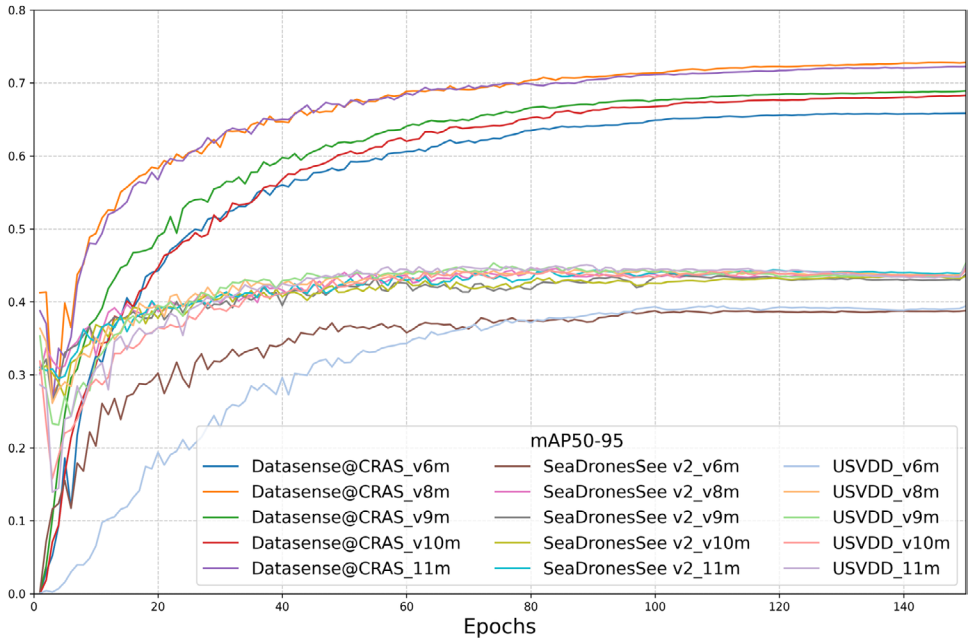

Fig. 3 Evolution by epochs during the training phase for the medium model in each version and dataset

Orin Nano counterpart but is significantly more expensive. In contrast, Jetson Orin Nano and Intel UP Square are better suited to IoT applications, with lower energy consumption requirements, as well as more balanced performance and cost. We would like to highlight that both NVIDIA boards offer various power consumption modes, allowing users to enable or disable certain cores and adjust frequency settings for adapting computational resources to the specific workload according to the user's requirements.

3.5 Evaluation metrics

The performance of an object detection model is measured mainly by the AP metric [21]. The main metrics include Precision and Recall, as defined in Eq. 1. Precision measures the proportion of correctly predicted positive samples among all predicted positive samples [42, 43], while recall evaluates how many of the actual positive samples were correctly identified [43]. Another key metric is mAP (Mean Average Precision), which averages precision–recall values at multiple intersection-over-union (IoU) thresholds to assess the overall performance of the model [44]

$$P = \frac{TP}{TP + FP} \quad R = \frac{TP}{TP + FN} \quad mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i. \tag{1}$$

3.6 Optimization techniques

Optimization in neural networks is used primarily to minimize criteria, such as energy consumption, bandwidth, and model size [45]. The study by [19] mentions recommended techniques that can contribute to developing more robust lightweight models; therefore, for this study, some of these techniques are described, which will be implemented in the proposed models.

The pruning technique consists of eliminating irrelevant neurons from the convolutional operations of the neural network to improve the efficiency and performance

Table 4 YOLO performance comparison using the Datasense@CRAS dataset: benchmarking across models of different sizes

Model	mAP50	mAP50-95	R	P
YOLOv6n	83.0	60.0	77.0	86.0
YOLOv6 s	86.0	65.0	80.0	89.0
YOLOv6 m	87.0	66.0	83.0	90.0
YOLOv8n	90.0	68.0	85.0	92.0
YOLOv8 s	91.0	71.0	86.0	93.0
YOLOv8 m	93.0	73.0	88.0	94.0
YOLOv9 t	85.0	63.0	80.0	87.0
YOLOv9 s	87.0	66.0	82.0	91.0
YOLOv9 m	89.0	69.0	84.0	93.0
YOLOv10n	84.0	62.0	77.0	88.0
YOLOv10 s	87.0	66.0	82.0	90.0
YOLOv10 m	88.0	68.0	82.0	91.0
YOLO11n	88.0	67.0	84.0	93.0
YOLO11 s	90.0	71.0	85.0	94.0
YOLO11 m	91.0	72.0	87.0	94.0

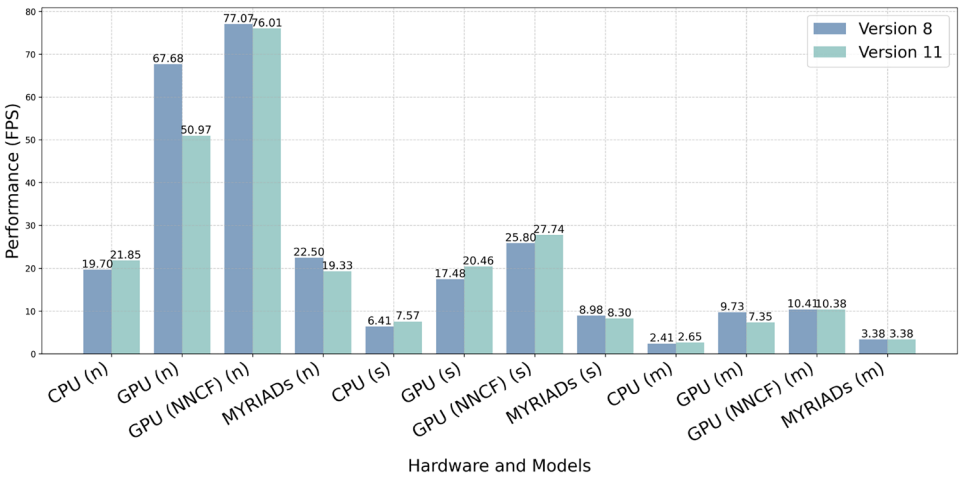

Fig. 4 Performance of each models with different computing processors on the Intel edge board. Note: The CPU supports precision of *fp32*, the GPU supports both *fp32* and *fp16*, and the VPU is limited to *fp16*

of the model [46]. Pruning can be applied before, during, or after training and can be classified as dynamic pruning or static pruning [47, 48]. The quantization technique, on the other hand, focuses on reducing the precision of the parameters forming the neural network, i.e., minimizing the number of bits, e.g., *fp32* to *int8*. This effect not only achieves greater compatibility with different hardware, but also improves inference speed, sacrificing minimal precision loss [46, 49]. Quantization can be applied through three different approaches: dynamic and static quantization (both post-training) and quantization-aware training. Each has its strengths and can be based on what one aims to achieve [50].

4 Experimental results

In this section, we present the experimental results, including analysis of performance metrics, the deployment of models on edge devices considering both performance and energy consumption, and comparisons with models from other studies and different versions of YOLO.

4.1 Performance analysis by epochs

Figure 3 shows the evolution of the mean variance epoch for the mAP50-95 metric for the three datasets during the training phase. Note that the best-performing models come from the Datasense@CRAS dataset for both version 8 and version 11.

Now, to make a more complete analysis, considering the Datasense@CRAS dataset for its better performance, Table 4 shows the evolution by epochs of the training versions for YOLOv6, YOLOv8, YOLOv9, YOLOv10, and YOLO11 using this dataset.

4.2 Testing the model on edge devices

4.2.1 Intel UP Squared Pro 7000 Edge

To deploy a YOLO model on Intel Edge devices (CPU, GPU, and VPU/NPU), it must be converted to the Intermediate Representation (IR) format using the OpenVINO framework. OpenVINO offers tools such as the Model Optimizer for model conversion and quantization tailored to the device's arithmetic, as well as the Neural Network Compression Framework (NNCF), which optimizes models through pruning, layer fusion, and precision reduction to *int8*. The model-optimized script (*mo.py*) available in OpenVINO can be invoked with the parameters *–compress_to_fp16=True –transform=Pruning* to apply pruning and compress the model from *fp32* to *fp16*. For NNCF optimization, the function *nncf.quantize* activates this improvement. However, the OpenVINO documentation includes more information about some of these features.[3]

Figure 4 presents the performance achieved on different processors and the improvement obtained by applying NNCF optimization, as well as quantization-aware pruning on the GPU. It is important to note that the size of the model significantly affects performance, as model *n* exceeds 50 fps on the GPU, while version *m* ranges between 7 and 10 fps. Note that for both versions 8 and 11, the GPU outperforms the CPU by more than 2× in both cases (NNCF and quantization-aware pruning enabled). Finally, a single Myriad accelerator achieves performance very similar to that of a multicore CPU. In this case, the models trained with the Datasense@CRAS dataset are evaluated.

[3] Convert and Optimize YOLOv8 with OpenVINO. https://github.com/openvinotoolkit/openvino_notebooks/blob/latest/notebooks/yolov8-optimization/yolov8-object-detection.ipynb.

Table 5 Performance of YOLO models on the AGX Orin under different precision arithmetic

Model	Prec.	version 8 Perf. (fps)	mAP	version 11 Perf. (fps)	mAP
n	fp32	333.87	0.88	241.37	0.88
	fp16	522.80	0.88	401.35	0.88
	int8	684.69	0.53	344.67	0.56
s	fp32	197.86	0.89	73.31	0.89
	fp16	361.50	0.89	149.82	0.89
	int8	498.81	0.65	204.28	0.66
m	fp32	102.35	0.92	150.15	0.90
	fp16	198.01	0.92	260.35	0.90
	int8	259.97	0.68	357.76	0.62

Table 6 Performance of YOLO models on the Jetson Orin Nano under different power modes

Model	Prec	version 8		version 11	
		Perf. (fps)		Perf. (fps)	
		at 15 W	at 7 W	at 15 W	at 7 W
n	fp32	108.18	48.49	73.55	34.4
	fp16	178.85	83.73	116.07	52.77
	int8	228.14	93.35	135.82	60.45
s	fp32	58.81	26.00	47.80	21.44
	fp16	106.29	47.12	81.90	38.13
	int8	165.24	68.69	117.77	52.82
m	fp32	27.92	11.28	22.67	9.58
	fp16	58.52	23.79	41.10	18.31
	int8	82.92	33.53	67.41	28.02

4.2.2 Nvidia Jetson AGX Orin

Table 5 shows the impact of the quantization of the model on the NVIDIA Jetson AGX Orin, evaluated with arithmetic in *fp32*, *fp16*, and *int8*. Performance, measured in frames per second (fps) with the models from the Datasense@CRAS dataset, reveals that quantization significantly improves speed, with an average increase of 1.8× for *fp16* and 2.4× for *int8*. Furthermore, as can be seen, the mAP for *fp32* and *fp16* arithmetic are similar, while for *int8*, the difference is noticeable.

4.2.3 Nvidia Jetson Orin Nano

The performance of the Jetson Orin Nano device, which can operate in either 7- or 15-watt modes, is evaluated. This configurability makes it particularly well suited for deployment in realistic IoT scenarios, such as battery-powered USV environments. Table 6 presents the impact of model quantization using *fp32*, *fp16*, and *int8* arithmetic on inference performance. The mAP values are not included in this table, as the accuracy results are identical to those reported in Table 5, based on the evaluation measurements conducted.

The results indicate that the Jetson Orin Nano system meets real-time constraints in the 15 W mode and achieves more than 25 fps in the *n* and *s* models in the 7 W mode. Furthermore, *int8* quantization improves the speeds in all variants.

4.3 Power consumption

A key criterion for the deployment of neural network models in real systems is energy consumption, especially in IoT devices with limited energy budgets. This section presents results on energy requirements.

4.3.1 Intel UP Squared Pro 7000 Edge

Figure 5 shows the energy consumption of the CPU, GPU, and the Intel MYRIAD accelerator. The inference of the

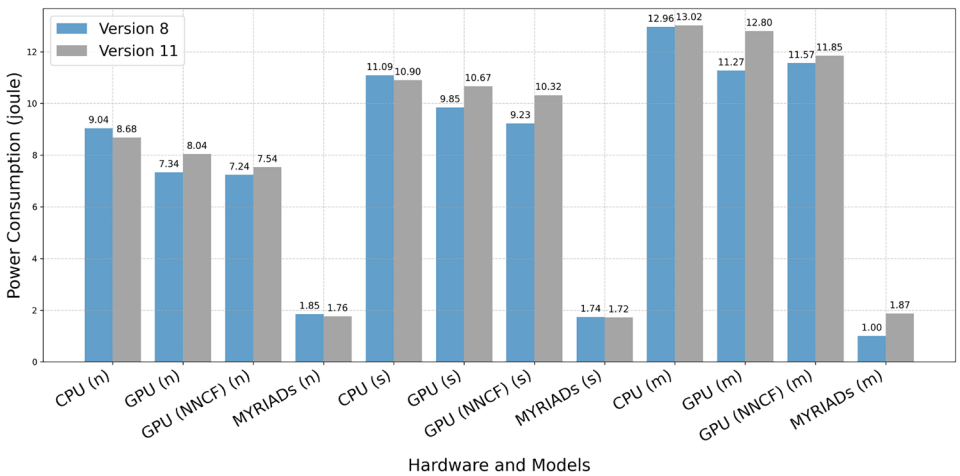

Fig. 5 Power consumption of YOLOv8 and YOLO11 on Intel edge computing processors across different arithmetic precisions

MYRIAD models indicates that it is the most efficient device, while the CPU exhibits the highest energy demand. However, the GPU shows intermediate behavior for both versions.

4.3.2 Nvidia Jetson devices

Figure 6 illustrates the power consumption in different boards and configurations. The NVIDIA AGX, designed for high-performance industrial environments, exhibits significantly higher power consumption compared to the Orin Nano, a low-power device.

The 7 W and 15 W modes of the Orin Nano were evaluated, with the 7 W mode being the most efficient. It meets real-time requirements in most environments, making it an ideal choice for USVs.

4.4 Comparison with other studies

To compare the proposed models, we used the Datasense@CRAS dataset. Table 7 includes relevant studies on ship detection or classification. Zhang et al. [51] trained their model with SeaShip7000 images, WSODD, and data from a USV equipped with a photoelectric capsule device at various water locations. Liu et al. [52] used the COCO dataset and other images downloaded to focus on the category of ships. Zhou and Peng [53] also used SeaShip7000 to train their model.

Table 7 shows that our models (YOLOv8 m-our and YOLO11 m-our) stand out with a precison of 94.0%, surpassing 91.57% reported by Zhang et al. [51], and achieving a mAP50-95 of 73.0%, compared to 58.87% in the same study. These results show a remarkable improvement in object detection at different IoU thresholds. In addition, our models show better retrieval performance and accuracy in versions 8 and 11 compared to Liu et al. [52] and Zhou and Peng [53], confirming their reliability and robustness in real-time detection. It should be noted that the models labeled with the term “original"in Table 7 refer to those downloaded directly from the Ultralytics website, and the tests were carried out using images from our dataset (Datasense@CRAS), providing a relevant comparison with the custom models we developed.

4.5 Comparison with other YOLO models

To validate the proposed models, in Fig. 7, we compare different versions of YOLO, using the Datasense@CRAS dataset to ensure a fair evaluation. We analyze performance, taking the average power consumption of the Nvidia Jetson Orin Nano in 15 W mode at 3-s intervals. The results highlight that the proposed models are more efficient, achieving lower inference time, better performance, and a slight increase in mAP50-95. In particular, YOLOv8n is positioned as the best choice for resource-constrained hardware, while YOLOv8 m is more suitable for higher accuracy and performance demands.

Table 7 Comparison of our models with those proposed in other studies using the Datasense@CRAS dataset as a baseline

Models	mAP50	mAP50-95	R	P
YOLOv8n-original	0.4	0.2	13.0	0.5
YOLOv8 s-original	0.5	0.3	10.1	0.6
YOLOv8 m-original	0.2	0.1	8.2	0.4
YOLOv8n-our	90.0	68.0	85.0	92.0
YOLOv8 s-our	91.0	71.0	86.0	93.0
YOLOv8 m-our	93.0	73.0	88.0	94.0
YOLO11n-original	0.7	0.4	11.3	0.8
YOLO11 s-original	0.7	0.5	7.9	1.1
YOLO11 m-original	0.9	0.5	8.3	1.7
YOLO11n-our	88.0	67.0	84.0	93.0
YOLO11 s-our	90.0	71.0	85.0	94.0
YOLO11 m-our	91.0	72.0	87.0	94.0
Zhang et al. [51]	92.85	58.87	90.23	91.57
Liu et al. [52]	56.6	26.8	–	71.9
Zhou and Peng [53]	88.6	62.1	86.5	80.8

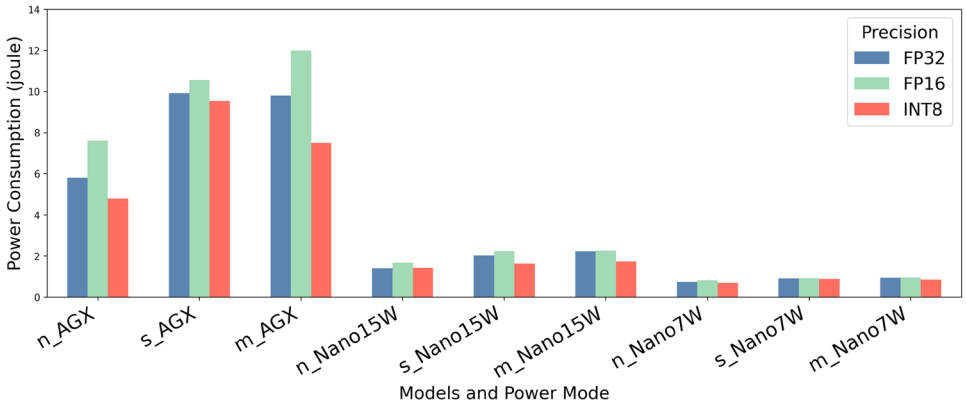

Fig. 6 Power consumption in AGX Orin and Orin Nano under different precision modes and power configurations

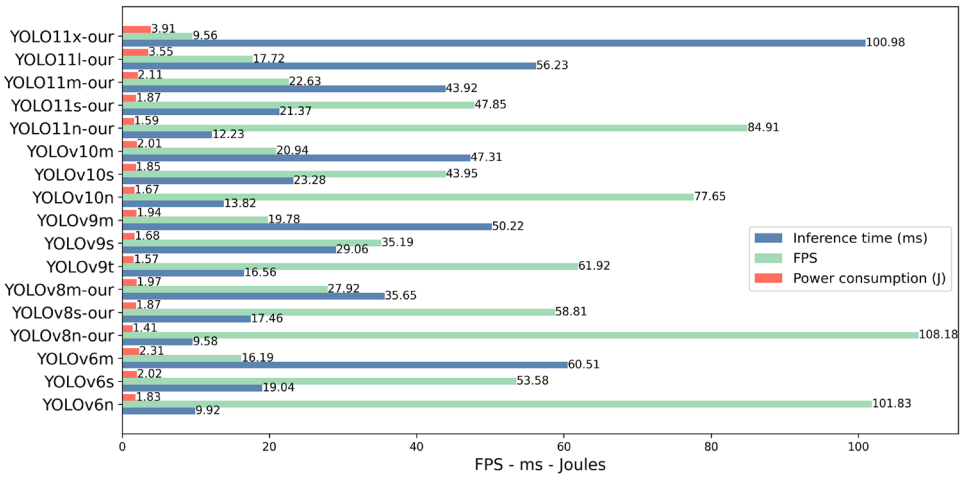

Fig. 7 Performance analysis of YOLO models in terms of inference time, power consumption, and fps

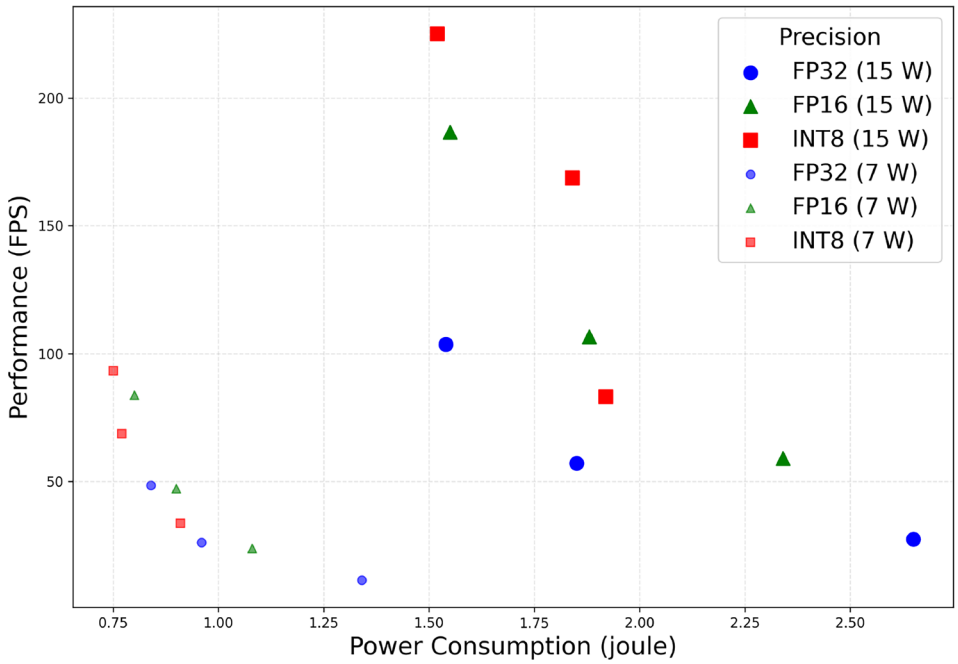

Fig. 8 Pareto trade-off: performance vs. power consumption

The results shown in Fig. 7 demonstrate that the models proposed in our study not only improve on previous versions of YOLO but also outperform the latest state-of-the-art (SOTA) models. The performance improvement of our models is because the versions YOLOv6, YOLOv9, YOLOv10, and YOLO11 have higher architectural complexity, which slightly reduces the performance. Furthermore, YOLOv10 [54], with its more complex design and two-stage training, has been observed to increase inference time and resource requirements, further affecting FPS compared to the single-stage approach of YOLOv8. In this context, Ranjan's study [55] established that YOLOv8 outperforms YOLO11 in fruit segmentation speed, although with a slight trade-off in precision. However, Muhammad's post [56] suggests that both versions exhibit comparable overall performance, with no significant differences in effectiveness.

5 Discussion

In this study, three IoT neural devices were evaluated to determine which offered the best performance-to-price ratio for the implementation of a vision system. The results show that while the NVIDIA Orin AGX offers the highest performance, its high cost—four to five times higher than other options—makes it unfeasible except in scenarios where real-time requirements cannot be met with cheaper alternatives. This study demonstrates that more affordable options, combined with optimizations, meet real-time requirements.

Therefore, Fig. 8 presents a Pareto chart analysis of the relationship between performance versus power consumption for each model for the Intel devices and the NVIDIA Jetson Orin Nano (7 W and 15 W).

On the one hand, models converted to *int8* arithmetic exhibit the highest efficiency ratio. However, performance is proportional to the configured power mode; therefore, the 15-watt power mode achieves faster inference while consuming, on average, twice as much power. Finally, we conclude that the vision system based on the NVIDIA Jetson Orin Nano device meets real-time requirements, with measured energy consumption below 1 watt for larger models such as *m* (arithmetic *int8*).

Fig. 9 Vision system deployment in a real USV

5.1 Video system deployment

In this section, we present the vision system developed to detect objects in the USV's trajectory and avoid collisions. The photograph in Fig. 9 shows the system assembled in the USV, which has been designed to collect water samples from reservoirs to analyze water quality and detect toxicity caused by cyanobacteria blooms. The image illustrates the acquisition of images using the ZED 2 camera and their subsequent real-time processing by the Jetson Orin Nano-based system. The remaining electronic equipment shown corresponds to the engine control system.

Finally, we would like to illustrate the object detection system integrated with the ZED 2 camera in a real scenario. Figure 10 presents a real image capture used for obstacle detection, which showcases the system's ability to identify objects in the scene and measure their distances. The detection of objects in this image was performed with the model that was trained with the USVDD dataset, since it is the one that has the classes in accordance with the scenario of the video where inference was performed.

Fig. 10 Real image obstacle detection in the vision system on the USV

6 Conclusion

This work aims to develop a system for detecting objects in the path of an aquatic USV designed to collect water samples for water quality analysis, thereby preventing collisions with potential obstacles. The requirements include implementing a vision system capable of real-time video processing with high accuracy in object identification and distance measurement while maintaining low energy consumption, as the USV is battery-powered.

The main findings of this research are as follows.

- *Optimization and quantization*: these methods significantly improve response times, enabling real-time performance without compromising accuracy. Notably, NNCF provides substantial speed enhancements without requiring additional programming by utilizing lower-precision options, such as half-precision floating point and integer representations.
- *Comparison of edge boards*: the Intel Edge device stands out for its versatility, allowing users to deploy models on a CPU, GPU, or VPU by modifying a single line of code. Additionally, multiple MYRIAD VPUs offer a performance-to-power ratio comparable to that of CPUs and GPUs. Although NVIDIA's Orin AGX delivers superior performance, its higher cost and power consumption make the Orin Nano a more suitable alternative.
- *Feasibility and efficiency*: when performance, energy efficiency, and cost are considered, the Orin Nano proves to be the better choice for real-time applications. In 15-watt mode, it can run the high-precision YOLOv8 m model on less than 3 watts, making it an effective solution for Unmanned Aquatic Vehicles.
- *Model comparisons*: our proposed models outperform recent YOLO architectures, demonstrating that the CNN-based version of YOLOv8 is more efficient for real-time operations close to the latest version of YOLO11.
- *Future work*: a preliminary analysis of YOLOv12-nano shows similar accuracy to YOLOv8, with only a slight 2% performance speed-up. A more detailed comparison with earlier versions is planned for the USV context.

Acknowledgements This paper has been partially funded by the EU (FEDER), the Spanish MINECO under Grant Nos. PID2021-126576 NB-I00 and TED2021-130123B-I00 funded by MCIN/AEI/10.13039/501100011033 and by European Union "ERDF A way of making Europe"and the NextGenerationEU/PRT. J.L.M. thanks the National Secretariat of Science, Technology and Innovation (SENACYT) of Panama for the financial support during the completion of his PhD.

Data Availability The datasets, trained models, and sample testing videos used in this study are publicly available at the GitHub repository https://github.com/artecs-group/Yolo-AquaticUSV, along with usage instructions.

References

1. Lin, Li., Liao, Xiaofei, Jin, Hai, Li, Peng: Computation offloading toward edge computing. Proc. IEEE **107**(8), 1584–1607 (2019)
2. Cisco, U.: Cisco annual internet report (2018–2023) white paper. Cisco: San Jose, CA, USA **10**, 1–35 (2020)
3. Georgiou, Kyriakos, Xavier-de Souza, Samuel, Eder, Kerstin: The IoT energy challenge: a software perspective. IEEE Embed. Syst. Lett. **10**(3), 53–56 (2017)
4. Liu, Dan, Yan, Zheng, Ding, Wenxiu, Atiquzzaman, Mohammed: A survey on secure data analytics in edge computing. IEEE Internet Things J. **6**(3), 4946–4967 (2019)
5. Chang, Zhuoqing, Liu, Shubo, Xiong, Xingxing, Cai, Zhaohui, Guoqing, Tu.: A survey of recent advances in edge-computing-powered artificial intelligence of things. IEEE Internet Things J. **8**(18), 13849–13875 (2021)
6. Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang et al. 2024. Imagenet large scale visual recognition challenge. https://www.image-net.org/. Accessed 19 Nov 2024
7. Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2024. Microsoft coco: common objects in context. https://cocodataset.org/#home. Accessed 19 Nov 2024
8. Geiger, Andreas, Philip Lenz, Christoph Stiller, and Raquel Urtasun. 2024. Kitti vision benchmark suite. https://www.cvlibs.net/datasets/kitti/. Accessed 19 Nov 2024
9. Gaugue, Alain, Menard, Michel, Migot, Etienne, Bourcier, Pierre, Gaschet, Clement: Development of an aquatic usv with high communication capability for environmental surveillance. *OCEANS 2019 - Marseille, OCEANS Marseille 2019* (2019)
10. Risco-Martín, José L., Esteban, Segundo, Chacón, Jesús, Carazo-Barbero, Gonzalo, Besada-Portas, Eva, López-Orozco, José A.: Simulation-driven engineering for the management of harmful algal and cyanobacterial blooms. Simulation **99**(10), 1041–1055 (2023)
11. Carazo-Barbero, Gonzalo, Besada-Portas, Eva, Risco-Martín, José Luis, López-Orozco, José Antonio: Ea-based asv trajectory planner for detecting cyanobacterial blooms in freshwater. In: *GECCO 2023 - Proceedings of the 2023 Genetic and Evolutionary Computation Conference*, pages 1321–1329. Association for Computing Machinery, Inc, (2023)
12. Ahmed, Shams Forruque, Sakib, Alam, Md., Bin, Hassan, Maruf, Rozbu, Rodela, Mahtabin, Ishtiak, Taoseef, Rafa, Nazifa, Mofijur, M., Shawkat Ali, A.B.M., Gandomi, Amir H.: Deep learning modelling techniques: current progress applications advantages and challenges. Artif. Intell. Rev. **56**(11), 13521–13617 (2023)
13. Deng, Changyu, Ji, Xunbi, Rainey, Colton, Zhang, Jianyu, Wei, Lu.: Integrating machine learning with human knowledge. iScience **23**, 101656 (2020)
14. Yamashita, Rikiya, Nishio, Mizuho, Do, Richard Kinh Gian., Togashi, Kaori: Convolutional neural networks: an overview and application in radiology. Insights Imaging **9**, 611–629 (2018)
15. Elshawi, Radwa, Wahab, Abdul, Barnawi, Ahmed, Sakr, Sherif: DLBench: a comprehensive experimental evaluation of deep learning frameworks. Clust. Comput. **24**(9), 2017–2038 (2021)
16. Ultralytics yolov8 docs
17. What is pytorch? | data science | nvidia glossary

18. Openvino™ toolkit overview - openvino??? documentation - version(archive)
19. Mittal, Payal: A comprehensive survey of deep learning-based lightweight object detection models for edge devices. Artif. Intell. Rev. **57**(9), 1–49 (2024)
20. Azevedo, Pedro, Santos, Vítor.: Comparative analysis of multiple YOLO-based target detectors and trackers for ADAS in edge devices. Robot. Auton. Syst. **171**, 104558 (2024)
21. Lema, Darío G., Usamentiaga, Rubén, García, Daniel F.: Quantitative comparison and performance evaluation of deep learning-based object detection models on edge computing devices. Integration **95**, 102127 (2024)
22. Di Fabrizio, Giacomo, Calisti, Lorenzo, Contoli, Chiara, Kania, Nicholas, Lattanzi, Emanuele: A study on the energy-efficiency of the object tracking algorithms in edge devices. In: *Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing*, UCC '23, New York, NY, USA, (2024). Association for Computing Machinery
23. Carrillo-Perez, Borja, Rodriguez, Angel Bueno, Barnes, Sarah, Stephan, Maurice: Improving yolov8 with scattering transform and attention for maritime awareness. In: *2023 International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 1–6, (2023)
24. Wang, Haochen, Shi, Juan, Karimian, Hamed, Liu, Fucheng, Wang, Fei: YOLOSAR-lite: a lightweight framework for real-time ship detection in SAR imagery. Int. J. Digit. Earth **17**(1), 2405525 (2024)
25. Liu, Yuxi, Wu, Yanliang, Ma, Zheng, Zhou, Yi, Li, Guoqiang, Jian, Xia, Meng, Shijin: CNN and binocular vision-based target detection and ranging framework of intelligent railway system. In: IEEE Transactions on Instrumentation and Measurement (2024)
26. Dong, Kaiyuan, Liu, Tao, Shi, Zhen, Zhang, Yang: Accurate and real-time visual detection algorithm for environmental perception of USVS under all-weather conditions. J. Real-Time Image Proc. **21**(4), 1–18 (2024)
27. Sohan, Mupparaju, Ram, Sai, Thotakura, Rami Reddy, Venkata, Ch.: A review on YOLOv8 and its advancements. In: Jacob, I.J., Piramuthu, S., Falkowski-Gilski, P. (eds.) Data Intell. Cogn. Inform., pp. 529–545. Springer Nature Singapore, Singapore (2024)
28. Wang, Hai, Liu, Chenyu, Cai, Yingfeng, Chen, Long, Li, Yicheng: YOLOV8-QSD: an improved small object detection algorithm for autonomous vehicles based on yolov8. *IEEE Transactions on Instrumentation and Measurement* (2024)
29. Huang, Zhongjie, Li, Lintao, Krizek, Gerd Christian, Sun, Linhao: Research on traffic sign detection based on improved YOLOv8. J. Comput. Commun. **11**(7), 226–232 (2023)
30. Kumar, Debasis, Muhammad, Naveed: Object detection in adverse weather for autonomous driving through data merging and YOLOv8. Sensors **23**(20), 8471 (2023)
31. Xiuyan, G.A.O., Yanmin, Z.H.A.N.G.: Detection of fruit using YOLOv8-based single stage detectors. Int. J. Adv. Comput. Sci. Appl. (2023). https://doi.org/10.14569/IJACSA.2023.0141208
32. Xiao, Bingjie, Minh, Nguyen, , Wei Qi, Yan: Fruit ripeness identification using YOLOv8 model. Multimed. Tools Appl. **83**(9), 28039–28056 (2024)
33. Islam, M.P., Hatou, K.: Artificial intelligence assisted tomato plant monitoring system-an experimental approach based on universal multi-branch general-purpose convolutional neural network. Comput. Electr. Agric. **224**, 109201 (2024)
34. Atfield, J.E., Yaraskavitch, L.R., Rand, E.T.: Kinetics experiments in ZED-2 using heterogeneous cores of advanced nuclear fuels. Ann. Nucl. Energy **121**(11), 36–49 (2018)
35. Abdelsalam, Ahmed, Mansour, Mostafa, Porras, Jari, Happonen, Ari: Depth accuracy analysis of the zed 2i stereo camera in an indoor environment. Robot. Auton. Syst. **179**(9), 104753 (2024)
36. Göncz, Levente, András, László Majdik.: Object-based change detection algorithm with a spatial AI stereo camera. Sensors (Basel, Switzerland), **22**(9), 6342 (2022)
37. Trinh, Linh, Mercelis, Siegfried, Anwar, Ali: A comprehensive review of datasets and deep learning techniques for vision in unmanned surface vehicles. arXiv (2024). https://doi.org/10.48550/arXiv.2412.01461
38. Ding, Caiwen, Wang, Shuo, Liu, Ning, Xu, Kaidi, Wang, Yanzhi, Liang, Yun . 2019. REQ-YOLO: A resource-aware, efficient quantization framework for object detection on FPGAs. 10
39. Shao, Zeyuan, Lyu, Hongguang, Yin, Yong, Cheng, Tao, Gao, Xiaowei, Zhang, Wenjun, Jing, Qianfeng, Zhao, Yanjie, Zhang, Lunping: Multi-scale object detection model for autonomous ship navigation in maritime environment. J. Mar. Sci. Eng. **10**(11), 1783 (2022)
40. Compressed version - files - nextcloud
41. An annotated and classified maritime dataset aimed at machine learning - conjunto de datos - ckan
42. Umer, Muhammad, Zainab, Imtiaz, Saleem, Ullah, Arif, Mehmood, Gyu Sang, Choi: Fake news stance detection using deep learning architecture (CNN-LSTM). IEEE Access **8**, 156695–156706 (2020)
43. Hicks, Steven A., Strümke, Inga, Thambawita, Vajira, Hammou, Malek, Riegler, Michael A.., Halvorsen, Pål., Parasa, Sravanthi: On evaluation metrics for medical applications of artificial intelligence. Sci. Rep. **12**(4), 1–9 (2022)
44. Fei, Du., Mo, Dandan, Ma, Tianbing, Fang, Jiaxin, Shu, Jinxin, Long, Jitao: Rigid tank guide fault detection algorithm based on improved YOLOv7. J. Real-Time Image Proc. **22**(1), 2 (2024)
45. Hawks, Benjamin, Duarte, Javier, Fraser, Nicholas J., Pappalardo, Alessandro, Tran, Nhan, Umuroglu, Yaman: Ps and qs: quantization-aware pruning for efficient low latency neural network inference. Front. Artif. Intell. **4**, 2 (2021)
46. Liang, Tailin, Glossner, John, Wang, Lei, Shi, Shaobo, Zhang, Xiaotong: Pruning and quantization for deep neural network acceleration: a survey. Neurocomputing **461**(1), 370–403 (2021)
47. Ganguli, Tushar, Chong, Edwin K.P..: Activation-based pruning of neural networks. Algorithms **17**(1), 48 (2024)
48. Cheng, Hongrong, Zhang, Miao, Shi, Javen Qinfeng: A survey on deep neural network pruning-taxonomy, comparison, analysis, and recommendations. arXiv (2024). https://doi.org/10.48550/arXiv.2308.06767
49. Quantizing models post-training - openvino^{TM} documentation - version(archive)
50. neural-compressor/docs/source/quantization.md at master . intel/neural-compressor ?? github
51. Zhang, Lei, Du, Xiang, Zhang, Renran, Zhang, Jian: A lightweight detection algorithm for unmanned surface vehicles based on multi-scale feature fusion. J. Mar. Sci. Eng. **11**, 1392 (2023)
52. Liu, Wenzheng, Chen, Yaojie. 2023. Il-yolov5: A ship detection method based on incremental learning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 14087 LNCS. pp. 588–600
53. Zhou, Weina, Peng, Yujie: Ship detection based on multi-scale weighted fusion. Displays **78**(7), 102448 (2023)
54. Hussain, Muhammad: YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection. Machines **11**(7), 677 (2023)
55. Sapkota, Ranjan, Karkee, Manoj. 2024. Comparing YOLO11 and YOLOv8 for instance segmentation of occluded and non-occluded immature green fruits in complex orchard environment. ArXiv. arXiv:2410.19869.

56. Munawar, Muhammad Rizwan. 2025. Pose models: ultralytics YOLO11 vs ultralytics YOLOv8. LinkedIn post. Accessed 14 Feb 2025