Future Generation Computer Systems 162 (2025) 107481

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Underwater Mediterranean image analysis based on the compute continuum paradigm

Michele Ferrari [a], Daniele D'Agostino [a,*], Jacopo Aguzzi [c,d], Simone Marini [b,d,e]

[a] Department of Informatics, Bioengineering, Robotics and Systems Engineering (DIBRIS), Università degli studi di Genova, Via Dodecaneso 35, Genoa, Italy
[b] National Research Council of Italy (CNR), Institute of Marine Sciences, Pozzuolo di Lerici, La Spezia, Italy
[c] Department of Renewable Marine Resources, Instituto de Ciencias del Mar (ICM-CSIC), Pg. Marítim de la Barceloneta 37, Barcelona, Spain
[d] Stazione Zoologica Anton Dohrn (SZN), Naples, Italy
[e] NBFC, National Biodiversity Future Center, Palermo, Italy

ARTICLE INFO

Keywords:
Compute continuum
Object detection

ABSTRACT

Human activity depends on the oceans for food, transportation, leisure, and many more purposes. Oceans cover 70% of the Earth's surface, but most of them are unknown to humankind. This is the reason why underwater imaging is a valuable resource asset to Marine Science. Images are acquired with observing systems, e.g. autonomous underwater vehicles or underwater observatories, that presently transmit all the raw data to land stations. However, the transfer of such an amount of data could be challenging, considering the limited power supply and transmission bandwidth of these systems. In this paper, we discuss these aspects, and in particular how it is possible to couple Edge and Cloud computing for effective management of the full processing pipeline according to the Compute Continuum paradigm.

1. Introduction

The total surface of the Earth covered by Oceans is more than 70% and international trades are run 90% by nautical transportation. Human activity depends on the oceans for food, transportation, leisure and, at the same time it mitigates the climate change by producing oxygen, absorbing CO_2 and balancing the heat exchange. Studying, monitoring, and understanding the marine ecosystem with its biodiversity, together with the human impact on it is therefore crucial as the vast majority of the seas are unknown to humankind [1].

Biodiversity is defined as the variety of life ranging from genes to entire ecosystems. It is also commonly referred as the richness of species in a specific region or site [2]. Here we specifically focus on monitoring biodiversity among fish that are found in the Mediterranean Sea. In this case, by monitoring, we do not mean the verification of a certain species living in an area. We rather focus on the more challenging task of the quantitative analysis of the presence of some selected species of interest, coupling this information together with the environmental parameters driving the spatial/temporal species dynamics.

This activity has several long-term goals. Especially for marine protected areas, keeping track of the presence in time of certain organisms within its bounds, supporting the assessment of the health status of the observed area. Monitoring life at sea is fundamental for environment conservation and sustainable economic growth, for communities living on islands or along the shorelines [3]. Due to the vastity of this environment, not only in terms of ground distance but of underwater depth, it is essential to leverage observing systems that can operate in remote or underwater locations.

Underwater imaging is a valuable resource asset to marine sciences [4–6]. Researchers deploy cameras to extract visuals and support their work in combination with other sensory. Images and biological time data can reveal precious insights into behavioral patterns of marine life [7].

Images can be acquired by underwater cabled observatories, which are permanent installations on the seafloor that mount oceanographic and biological sensory. These installations often have cameras to record video footage or to take photos at regular intervals. Depending on the available bandwidth, the acquired data can be transferred back to land, using the cable to which the observatory is attached. It is the case of the regional cable observatory NEPTUNE. It is a network of observatories in the North East Pacific Sea, between USA and Canada [8]. Another fixed cabled observation point is the Observatory of The Sea in the Western Mediterranean Sea [9]. Permanent marine observatories cannot necessarily be cabled. In this case, they transmit data through a radio connection to a land station as in the case of the Acqua Alta Oceanographic Tower in the Adriatic Sea [10].

* Corresponding author.
E-mail addresses: daniele.dagostino@unige.it (D. D'Agostino), simone.marini@cnr.it (S. Marini).

https://doi.org/10.1016/j.future.2024.107481
Received 28 November 2023; Received in revised form 2 August 2024; Accepted 10 August 2024
Available online 12 August 2024

Other sources of images can be underwater devices as Autonomous Underwater Vehicles (AUV) and Remotely Operated Vehicles (ROV), which are underwater tethered robots remotely controlled. Example of AUV are the Autosub6000 AUV, which has been deployed to provide biological and habitat insight by taking pictures of the seabed with a high-resolution camera [11] or the ENDURUNS system, consisting of an Unmanned Surface Vehicle (USV) and an AUV, the latter equipped with camera echosounder sensors [12,13]. These devices can expand their monitoring to different areas either operating autonomously, attached to a boat, or to an underwater observatory.

Smaller autonomous apparatuses are also being used. These are generally equipped with the bare minimum to operate a photo camera, have battery power, and are relatively cheap, small, and flexible enough to be deployed on already existing oceanographic sensory out in the sea. An example is the macro gelatinous zooplankton monitor device GUARD-1 [5,6,14] for monitoring the underwater macro and mega fauna.

All these image capture devices, from the larger observatories to the small and energy-efficient instruments, can be used to explore larger areas of the oceans in combination with global networks like the drifter buoy array or the Argo Float program [14–16]. The consequence is that the amount of imagery is overwhelming and increases every year [17]. Elaborating those images by means of human domain knowledge is challenging and time-consuming.

The first aspect to consider, most of the time, is represented by properly managing the acquisition step in such constrained environments, where lack of transmission bandwidth and power supply are key matters. The second aspect is related to the specific requirements for the analysis flow, mostly based on Computer Vision (CV) and Artificial Intelligence (AI) methods. Underwater scientific sensors and devices operate in the constantly changing depths of the sea. Content-based knowledge extraction of the environment is crucial for making valuable scientific observations. For example, if the detection of worthy species from imaging is available underwater, an AUV can act according to what is the purpose of the exploration.

It is therefore clear that the traditional Cloud-centric model, where all images are periodically transferred to a remote computing center for full processing, is no longer feasible. There is the need to design new systems capable to seamlessly execute and relocate the component of the analysis pipeline along a continuum of resources spanning from the image-capturing device to the Cloud. This approach is widely used in many application contexts, e.g. industry 4.0 [18], medicine [19], metagenomic analysis [20], smart agriculture [21] and smart cities [22].

The resulting applications should optimize power and bandwidth consumption, carefully balancing the energy required to complete a specific task. Likewise, they should be able to manage data transmission in a smart way, balancing pros and cons of preprocessing the images in situ and transmitting only those with relevant content, avoiding battery waste. This represents the goal of the paper, i.e. an image analysis pipeline designed in accordance with the compute continuum paradigm in the field of biodiversity monitoring.

In detail, we discuss an end-to-end pipeline to detect, classify, and count fishes from underwater image acquisitions in the context of the continuum between onboard edge devices and remote cloud resources for storing, fusing, and analyzing the acquired data. In this pipeline the Cloud gathers heterogeneous data from different sources at the edge of the system (e.g. a network of underwater observatories). These data are integrated and further analyzed to extract knowledge, otherwise not accessible from a single sensor/observatory. The implementation will focus on the pipeline's edge side, as we are interested in a future underwater deployment. Therefore, we will present an experimental evaluation of the achievable performance in terms of execution time, power consumption, and accuracy in order to select the best configuration on the basis of different operative conditions. This analysis has a general value because the same approach should be applied to scenarios with similar characteristics, i.e. many raw data, limited power and bandwidth.

The paper is structured as follows. Section 2 presents related works. Section 3 gives an overview of the analysis pipeline. Section 4 describes the achieved experimental results, followed by a discussion of the applicability in different operative conditions and the Cloud component respectively in Sections 5 and 6. Conclusions and future directions are outlined in Section 7.

2. Related works

The efficient and effective integration of Cloud, Edge, and IoT technologies represents one of the key challenges of the next years in different domains, both for scientific and industrial applications [23]. The main reason is the explosion of data volumes generated by an increasing number of IoT devices, which requires an evolution of distributed digital infrastructures for storing, managing, and processing them.

Several review papers, books, and conference proceedings have been recently published on this topic. For example [24] discusses the differences between the definitions, while [25,26] analyses state-of-the-art solutions in architectures and communication protocols. Other papers focus on specific aspects as resource management [27], workflow management [28], available middleware [29], security and privacy issues [30], and on distributing intelligence in these scenarios [31].

From the analysis of these papers, we agree on the fact that "The compute continuum is an extension of the traditional Cloud towards multiple entities that provide analysis, processing, storage, and data generation capabilities" [24], and that edge computing lacks standardization of many aspects, and in particular of development guidelines [32].

While some domains are ahead in considering this paradigm for real-world application, as industry 4.0 [18,33], for marine science this represents a novel approach, as we will discuss in more detail in Section 6. According to the classification provided in [34], the compute continuum solution we are presenting belongs to the mobile Cloud computing class. The edge component focuses on classifying the elements of underwater images for monitoring purposes (e.g. to communicate early warnings) and to reduce network load by sending only relevant images for further processing steps to the Cloud. The Cloud component is responsible to store and analyze them in more details by aggregating heterogeneous data from different sources and/or with compute-intensive workloads.

For this reason, in the following, we focus on related works for image analysis in the marine domain.

This field is highly related to the development of Convolutional Neural Networks (CNN) [1,35]. We will focus on papers regarding fish classification and detection from underwater imaging. Moreover, we will distinguish between methods based on traditional CV algorithms and Deep Learning-based approaches (DL).

One of the first works is represented by [36] and consists in a texture-based classification. The goal was to distinguish between two visually similar fish species characterized by a strong hue that dominates the color information of images. The features used were built with edge extraction, performed with the well-known Canny algorithm, and Support Vector Machines (SVM) were used for specimen discrimination.

In [37] an automatic fish classification framework for underwater species based on discriminant analysis has been proposed consisting of two subtasks. The first one aims at extracting the organism trajectory along the video frames, based on Gaussian Mixture Model (GMM), Moving Average algorithm, and an Adaptive Mean Shift Algorithm. The second task extracts feature from the gray level histogram, Gabor filters, and gray level co-occurrence matrices for texture. Morphological operations are used to extract the contour of each fish which are then be used to compute a Curvature Scale Space (CSS). The CSS represents the zero crossings of the fish's evolution in the plane. A Principal

Component Analysis is run among both texture and boundary features before the Discriminant Analysis. The Discriminant Analysis consist of assigning a query observation x to a subset (class) of the whole feature set X. A K-Fold cross validation is used in order to decide how many subsets leave for testing out of the training set.

In [38] several classifiers based on Haar Cascades features for underwater fish detection have been employed and trained with the AdaBoost algorithm for classification.

The three previous works considered only a few images: the largest dataset has been considered in [38] and consists of 921 annotated underwater images taken in the Southern California Bight.

More recently the use of DL and larger datasets, as the Fish4Knowledge dataset [39], have been considered by the marine science community. Methods based on DL have been employed in a wider set of marine science applications, including harvested fish identification, fishery surveillance, and deep-sea mineral exploration [35]. In particular, Convolutional Neural Networks (CNN) have been adopted in order to tackle CV problems like image segmentation, object localization and counting.

In [40] a CNN network trained with large datasets extracted from Fish4Knowledge has been presented. In particular, the LifeCLEF 2014 image dataset (www.imageclef.org/) contains roughly 20k fish targets spread in about 1000 videos, counting a total of 10 observed species. A second set of images comes from LifeCLEF 2015, which raises the observed species to 15 while having a total target count larger than 20,000. A comparative study using other classic ML solutions like SVM and KNN have been performed by the authors, proving that the CNN approach as more stable and reliable in order to achieve a fully automated fish recognition process.

In [41] a 4 stage classifier based on 2 convolutional layers, a pooling layer for feature size reduction, a *Spatial Pyramid Pooling* (SPP) to extract information invariant to large poses has been proposed. The final classification step of the framework is a SVM obtaining classification performance higher than 90%.

Considering the high quality results obtained by CNNs in general purpose datasets like ImageNet and VOC [42] developed a framework specific for fish detection using a *Fast Region based Convolutional Neural Network* (Fast R-CNN). Experimental results were assessed using *mean Average Precision* (mAP) on 30,000 images extracted from Fish4Knowledge, with 12 species considered.

Another solution that leverages regions like CNN can be found in [43]. Their proposal is a three-stage pipeline that takes as input the combination of an underwater image, its Gaussian Mixture Model, and its Optical Flow (OF). The authors reported an accuracy of 87.44% claiming one of the best accuracies on LCF2015 at the time of publication.

Beside these domain-specific works, we have to consider the single-stage detector *You Only Look Once* (YOLO) [44], which rapidly gained momentum in the Computer Vision and AI community including the marine science community. The main advantage compared to RCNN is that YOLO addresses detection as a single regression problem from the input image to the predicted bounding box and class [44].

In [45] a modified version of YOLOv3 was exploited in order to obtain good results in varying seabed scenarios and in presence of crowded scenes. Its main contribution was to change the up-sample step and the addition of a SPP module at the end of Darknet-52. The latter is the backbone network of YOLOv3.

In [46] the authors worked on recognition of various seabed organisms like scallops, echinus, and starfish using a modified version of YOLOv4. Their proposal was to remove the up-sampling part of the network and replace it with a deconvolution in order to maintain the ability to restore details of the features after the convolution operator. The results are compared with other detector networks such as SDD and other versions of YOLO. In particular, authors put emphasis on the reduced parameters of their YOLOv4 variation from 64M to 16.7M.

A recent Biodiversity study of [17] leveraged YOLOv5.

Their work was carried out with manually tagged frames acquired from the NEPTUNE cabled observatory out the coasts of Canada. The overall annotated ground truth dataset is composed by 3647 images. The work is focused at detecting sablefish on different sites of the underwater cabled observatory and then estimate abundance. The YOLO object detector is then trained on a single target class. A first training is performed on the general purpose COCO dataset, and later a fine tuning is carried out on the annotated sablefish images. The obtained average precision on test data is reported to be 92%.

At last, we review some existing Edge Computing applications in object detection at sea. It is worth to consider that at present there is still a little number of these contributions, highlighting the relevance of the results presented in the present paper.

An interesting, general purpose work, focusing on the detection and classification of surface marine objects, such as passenger vessels, commercial vessels, and other marine floating objects has been published in [47]. They compared the use of YOLOv4 (64M parameters) and its optimized tiny version on the Nvidia Jetson Xavier AGX Edge Computing platform. The authors, although providing a few details about execution time and images used during tests, report a substantial increase of nearly 2x in inference speed when using YOLOv4 Tiny and a displacement of .04 in Mean Average Precision (mAP) on the Jetson Xavier AGX with respect to YOLOv4.

In [48] an application of CNNs with less than 20,000 parameters for fish detection in underwater environments has been proposed. It consists of a fixed, undersurface, observatory that leverages acoustic and optical sensors to activate a camera that acquires photos within 10 m field of view. This work focused on six different kinds of pelagic fishes, observable in the Mediterranean Sea. These images are processed with a multi-stage pipeline in order to give final predictions. As soon as the image is acquired from the camera, it goes at the segmentation stage which is performed by a specifically trained version of RetinaNET. These areas of the acquired image are the inputs for the subsequent stages. The object detection network has been designed with 4 blocks made by a Convolution layer of 16 filters with a kernel size of 3×3 followed by a max pooling layer. Then two dense layers of either 8, 16, or 32 units follow. The final dense layer outputs the binary choice. The multi-class classifier follows the same skeleton but with a $N_{classes}$ units final dense layer. To boost performances both in binary and multiclass classification, a tracking process between frames is added to follow the trajectory of each possible target. The algorithm used is SORT which has good performances in dealing with typical fish moving patterns. Data was manually annotated from several sources of images, including videos. The final count of segments of interest was 1.5M sparse in 51,000 images with which the classifiers were trained. The authors compared their work with a baseline network, VGG16. The authors reported comparable results with their baseline architecture, especially when the tracking algorithm was used to support classification. The deployment uses multiple digital cameras with one computing module each. The Edge Computing platform is an Nvidia Jetson TX2.

In [49] an object detector specifically designed for mobile marine platforms was presented. The authors proposed a modified version of YOLONano, with two distinct architectures, namely ULO and ULO Tiny. The main difference between ULO and ULO Tiny, is that the latter has been stripped off the third detection head. Moreover, ULO has an optional pre-processing module, to enhance the input image and achieve better results. The authors trained the detector on URPC2019, an underwater collection of seabed photos in the Chinese sea. In particular, it contains seabed images of scallops, echinus, and starfish acquired by human divers. The main goal of this tool is to support aquaculture applications with the use of a fast and efficient object detection architecture. They employ an optimized one-stage detector based on YOLO that can predict the target type and position with just one single architecture. The basic building block of ULO and ULO Tiny is the Ghost Module, which computes pseudo feature maps by employing cheap linear operations. The two architectures have

Table 1
The most significant classes among the 30 of the OBSEA imagery.

Class	#
Diplodus vulgaris	14 328
Oblada melanura	6 898
Diplodus sargus	2 772
Chromis chromis	2 762
Spicara maena	1 826
Coris julis	1 589

been directly mutated from YOLONano, substituting EP and PEP layers with either Ghost Bottlenecks or Ghost Modules. Moreover, they use a decoupled head design, splitting bounding box and class prediction into two separate branches. The reduction in terms of trainable parameters is considerable: YOLOv3 with image dimension of 512 × 512, counts up to 61.5M parameters, YOLONano 6.3M. ULO and ULO Tiny respectively 3.9M and 3.4M. The authors report ULO to perform 1.3% less than YOLOv3 in terms of mAP and outperforming other architectures like YOLOv4 and YOLONano. ULO and ULO Tiny were tested on the Nvidia Jetson Nano platform, showing promising runtime rates in terms of FPS, respectively 5.11 and 6.73.

3. The image analysis pipeline

In this Section we describe the component of the pipeline for underwater image analysis.[1]

At first, we describe the data selection and preparation we made for training and testing. Then we present the object detection stage. All the major numerical computations are implemented with the very well established torch framework, while specific utilities for dealing with image tensors are used from torchvision.

3.1. The dataset

The clearest trait that the related works had in common is the type of fishes on which they focused. In fact, most methods in the literature were trained, evaluated, and tested on datasets extracted from Fish4Knowledge.

Nevertheless, Fish4Knowledge contains tropical fishes, while our focus is on Mediterranean species listed in Table 1 and shown in Figs. 1 and 2. Moreover, Fish4Knowledge contains images taken only during daylight and in tropical waters totally different from what we observe in the Mediterranean Sea. In fact, the Mediterranean sea is rich in nutrients that actively influence the clarity and hue of the water (e.g. phytoplankton). Oppositely tropical waters are oligotrophic and hence clearer [50].

For these reasons, we used images produced with OBSEA, a cabled seafloor observatory located 4 km off the Vilanova i la Geltru coast in a fishing protected area [51]. The observatory is located at a depth of 20 m on a seabed composed of *Posidonia oceanica* in a fishing restricted area. The *Field of View* is 3 × 3 for a total reported 10.5 m^3 of imaged volume. The images have a resolution of 640 × 480 and are all JPEG encoded.

Across two full solar years, 2013 and 2014, a total number of 33,805 images have been captured. The total count of manually tagged fish organisms is of nearly 70,000 [9]. Each tagged fish has a *taxa* and a bounding box.

The data however is not immediately ready to be used in an object detection task with the most common methods we reviewed before. The abovementioned annotations of the bounding box come in the form of rotated rectangles for most objects. All the common object detection architectures work with bounding boxes that are parallel to the image axis.

[1] Code will be released via GitHub in case of acceptance.

This is the reason why a data preparation step is necessary in order to convert the rotated figure using an enclosing rectangle. This procedure, however, has the drawback of distorting the original dimension.

We addressed the matter by introducing a small translation for each exceeding box. As we work with models of the YOLO family, we convert these boxes in the format of x_c, y_c, w, h. The first two terms represent the center points for height and width of the box, while the last two correspond to the normalized height and width with respect to the image. The entire procedure is described in Algorithm 1, where λ is the translation factor for each exceeding bounding box.

Algorithm 1 Bounding Box transformation from a rotated rectangle to the standard normalized format of YOLO

```
Require: ${(x_1, y_1), ..., (x_4, y_4)}, (w, h), \lambda$
Ensure: $\mathbb{R}^4$
  $(x_{min}, y_{min}) \leftarrow \min(x_i, y_i)$
  $(x_{max}, y_{max}) \leftarrow \max(x_i, y_i)$
  if $x_{max} \geq w$ then
      $x_{min}, x_{max} \leftarrow x_{min} - \lambda, x_{max} - \lambda$
  end if
  if $y_{max} \geq h$ then
      $y_{min}, y_{max} \leftarrow y_{min} - \lambda, y_{max} - \lambda$
  end if
  $cx \leftarrow \frac{x_{max}+x_{min}}{2w}$
  $cy \leftarrow \frac{y_{max}+y_{min}}{2h}$
  $bw \leftarrow \frac{x_{max}-x_{min}}{w}$
  $bh \leftarrow \frac{y_{max}-y_{min}}{h}$
  return $[cx, cy, bw, bh]$
```

Among the entire OBSEA data, we reduced the classes by selecting the most relevant as in Table 1. We removed the data tagged with label *Unknown* as not relevant with respect to our task. That resizes the data to a total of 9613 images with a total of 30,130 fish annotations.

3.2. Training the object detectors

For the purpose of assessing the inference time and power consumption performance of object detection on an edge computing device, we trained a state-of-the-art YOLOv3 architecture alongside two computationally lighter models, ULO and ULO Tiny. The training phase has been conducted on a commercial Cloud platform specifically designed for Data Science purposes, leveraging an Nvidia GPU A100. It is well assessed that training could not happen on the edge device due to its computational resources which are not suitable for such task.

There are no pretrained checkpoints for either ULO nor ULO Tiny available and, for time purposes, we did not pre-train any of them. On OBSEA data a full training of ULO required less than 5 h, until early stopping. The data is split into three folds for training and validating the model during training epochs, leaving one out for the later testing phase on the Edge Computing device. We set the image dimension to 512 × 512 and keep fixed the batch size, learning rate, and optimizer for all the models.

Anchor selection. An improvement introduced in YOLOv2 is the bounding box prediction as displacements from anchor boxes. The height and width of each bounding box are computed as offsets from the anchor centroid. These boxes represent an input parameter for YOLO-like models. During the learning phase the network is able to adapt to randomly selected anchor dimensions, but starting from better priors leads to a smoother training phase [52]. We compute anchor centroids looking at the dimensions of bounding boxes in the training data, by selecting values that give a valuable representation of the objects that we have to work with.

We compute anchor centroids by running a *k-means* algorithm on all the bounding boxes within the training set. As we are interested

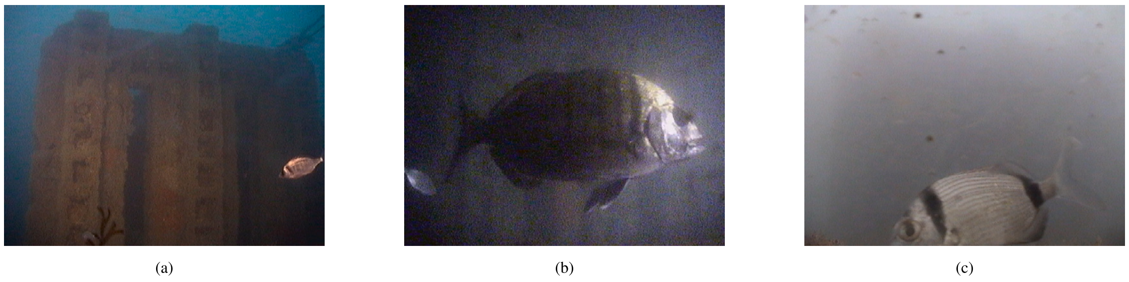

Fig. 1. The dataset contains a good amount of different capturing scenarios. We observe diverse water and light scenarios with varying capturing aspects and distances from the camera.
Source: Images from [9].

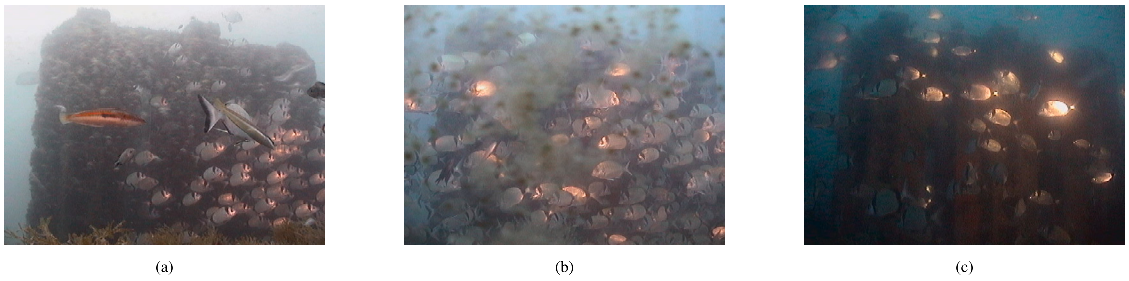

Fig. 2. Crowded scenes are also contained in the dataset, in varying light conditions.
Source: Images from [9].

in obtaining anchors that lead to small errors, we run the clustering using Eq. (1) where IoU is the *Intersection Over Union*, b being the tuple $(width, height)$ and c the centroid. IoU being independent of the size of the box make it better approach compared with the Euclidean distance.

$$d(b,c) = 1 - IoU(b,c) \tag{1}$$

Algorithm 2 K-Means Clustering Algorithm with Equation (1) as its distance metric

```
Require: $X \leftarrow \{b_1, b_2, ..., b_n\}$, $K$
Ensure: $C$
  $C \leftarrow \{c_1, ..., c_K\}$                ▷ randomly sample K centroids from X
  repeat
      for each $b_i \in X$ do
          $C_i \leftarrow C_i \cup \arg\max_j 1 - \text{IoU}(b_i, c_j)$     ▷ cluster assignment
      end for
      for $j$ from 1 to $K$ do
          $c_j \leftarrow \frac{1}{|C_j|} \sum_{b_i \in C_j} b_i$          ▷ cluster centroid update
      end for
  until when the centroids no longer change significantly
  return $C$
```

Once Algorithm 2 is run, we distribute equally the anchors to the 3 predictors of each model, and the K is set to 9. The centroids correspond to $(41,33)$, $(49,41)$, $(60,46)$, $(58,63)$, $(81,64)$, $(76,88)$, $(109,100)$, $(151,150)$, $(283,253)$.

Data augmentation. Early training attempts did have a very high validation loss, leading to poor generalization of unseen data. We thus adopted data augmentation as a regularization process during training. Augmentation was applied on the training data split, representing 80% of the whole data, i.e. 7695 images.

Due to the high volume of data and memory limitations of the training environment, augmentation could not be performed on the fly. Rather we built a second dataset that was merged with the original one and sampled together at every batch during training. The merge was performed with the data library of torch.

The augmentation process proportionally augments those classes underrepresented among those considered from the OBSEA dataset. We therefore set a limit of transformation each image can go through as well as the already higher represented target types. We applied the common image transforms:

- randomly performed horizontal flip;
- randomly performed vertical flip;
- affine transformation with an angle ranging from −25 to 25.

As the affine transforms rotates the image by an angle θ, the corresponding bounding boxes were adjusted as well. The new augmented dataset brought the total available images to 36,670 with a total of 62,796 annotated fishes.

Fish detection pipeline. Here we detail the full object detection pipeline that is going to be executed on the Edge Computing device. The requirement is a procedure that reads an image file from the device storage, runs a step of preparation, feeds the input through the object detector, and then outputs a $N \times 6$ matrix where N is the total number of fishes detected and, for each of them, 6 values:

- the class;
- the confidence score;
- a vector of 4 elements containing a bounding box prediction in the form of $(x_{min}, y_{min}, x_{max}, y_{max})$.

We use a *Non Maximum Suppression* (NMS) algorithm in order to filter the object detector predictions. It is presented in Algorithm 3. NMS selects the relevant region of interest by initially discarding all the boxes associated with a score less than the threshold we set. Then compare each and every box together, if a pair has an overlap larger than a threshold we compare their confidence and proceed to discard the less scoring bounding box. This is the same step used in Section 3.2 to evaluate the models during training. We run NMS in per-class batches of predictions. The final procedure:

- acquires the image in the input format;
- resizes to 512 × 512 pixels;

- runs the object detector;
- filters the output with batched NMS;
- converts bounding box format from (x_c, y_c, w, h) to $(x_{min}, y_{min}, x_{max}, y_{max})$;
- returns bounding boxes and fish type counts.

Algorithm 3 Non Maximum Suppression algorithm. B is the bounding box matrix, S confidence scores, λ IoU threshold and σ confidence threshold

```
procedure NMS(B, S, $\lambda$, $\sigma$)
    $K \leftarrow \emptyset$                                    ▷ indices to be kept
    for $i$ from 1 to $\#B$ do      ▷ exclude boxes with confidence lower than $\sigma$
        if $s_i \in S \geq \sigma$ then
            $K \leftarrow i$
        end if
    end for
    $T \leftarrow K$
    for $i \in T$ do
        for $j \in T$ do
            if IoU$(b_i, b_j) > \lambda$ then
                if $s_j > s_i$ then
                    $K \leftarrow K - i$                    ▷ discard box
                end if
            end if
        end for
    end for
    return $K$
end procedure
```

3.3. Monitoring framework

We now describe the monitoring framework used during the training and testing steps. In order to complete what has been described so far, we used three different computing environments: a development laptop, where all the code was developed, a Cloud computing service to train object detection models and collect the results, and the edge computing device used for the experimental part.

Thus we developed a monitoring system that collected all the required data and synchronized to the development machine via a local network. For such purpose, we used the *mlflow* [53] library that provides utilities to store machine learning experiments in a database and later access the data either with a user interface or programmatically.

We dealt with three types of measurement scenarios:

- punctual measurements: these are fixed numbers that corresponded to direct computations, for example, testing metrics;
- time variant measurements: these were measurements sampled over time, for example, the model training history, evaluation history, and energy consumptions;
- timers: these are values that are computed after a certain amount of time, for example all system related metrics presented in the experimental results Section.

We used an object we called *Logger*, that was passed around as a parameter to where logging was needed. All the measures acquired were then sent to the central mlflow repository via network with two modalities. In scenarios where the network overhead did not impact the monitoring, the values are immediately sent over to the central storage. During benchmarking a memory cache stores data and later sends the entire batch.

In the experimental results Section we will present costs in terms of runtime of the whole object detector pipeline. We used a different monitoring setup as the interest was even investigating how long the startup time weighted the total execution time. Onboard the edge computing device a thread runs a Logger object, with Python's *subprocess* module from the standard library, launches the object detection pipeline as a septate process. From the monitoring thread, we used the timer feature of Logger to compute the total runtime of the executable, comprehensive of the Python interpreter start time. On the subprocess a normal Logger runs and monitors metrics, logging also the metrics that are due from timers. The monitor thread reads all the data logged inside the subprocess from a file and then synchronizes back to the central repository.

Monitoring power consumption. A substantial part of the monitoring framework on the Edge computing device deals with power consumption. The Nvidia Jetson Nano we used has an *INA3221* chip that provides voltage and current readings. Such a chip works on the internal hardware bus, with the *I2C* protocol. The Python module *jetson-stats* [54] provides the necessary functionalities to interact with such the hardware consumption sensor, without the need to write raw serial communication code.

Values from the voltage/current sensor are sampled on a separate thread. We set the sampling frequency to be of 10 ms as it needs to be faster than the average inference time of the models we are monitoring. With such sampling speed, we are able to capture a clearer energy consumption pattern. Each time an image is evaluated from the object detector, we collect a reading. Finally, these values are stored as time series by the monitoring framework.

3.4. Result management

The present work represents an evolution of current monitoring devices, e.g., the one described in [55], based on the European patent EP2863257 [56].

These and most of the existing underwater sensors are responsible only for the image acquisition. Data are then sent, directly or via a base station, to a remote infrastructure, responsible for the classification, storage and provisioning of the most interesting images.

Our pipeline aims at moving part of the computation on the edge, represented by the monitoring devices, of distributed marine research infrastructures having the goal of integrating a variety of observing platforms and technologies to observe and monitor the marine environment [3,57,58].

With this approach, according to the observed environmental conditions, it will be possible to autonomously (a) identify, select, and integrate relevant information from a heterogeneous set of acquired data and, based on the results (b) activate/deactivate sensors, change the sampling frequencies and configurations.

It is therefore clear how the compute continuum paradigm represents a key aspect of developing effective, intelligent services that are not limited to image transmission. In this perspective, possible security issues should be considered and tackled with proper security mechanisms that handle the authentication of on-field devices and the confidentiality of the transmitted data [59].

4. Experimental results

In this Section, we provide an overview of the results we obtained from running an object detection pipeline on a Jetson Nano device.

In particular, we are going to present the results of three object detection models, ULO, ULOTiny, and YOLOv3 not only in terms of their achieved Mean Average Precision but also their runtime performance and consumption. The latter represents one of the major interests of this work.

4.1. The edge device

Nvidia Jetson boards have already been used for fish detection using underwater imaging, because they represent a very suitable solution for edge computing considering all together its low price, limited power consumption, and considerable compute capabilities provided by the CUDA architecture. Here the focus is on investigating the achievable performance of the previously described computer vision pipeline, with the aim to deploy and operate it on underwater observatories, AUVs, and ROVs in order to better support marine research.

Jetson Nano represents the *entry level* board in the Jetson Single Board Computer (SBC) range. The relevant hardware specifications are:

- 128-core NVIDIA Maxwell GPU @ 922 MHz;
- quad-core ARM A57 CPU @ 1.43 GHz;
- 4 GB 64-bit LPDDR4;
- hardware interfaces such as GPIO and I2C[2] and MIPI CSI-2.

Software. It is shipped with a pre-built Ubuntu 18.04 Operating System (OS) image with the foundation needed to run AI-centric applications. The system image is provided with Nvidia's proprietary SDK called JetPack. Nvidia JetPack is a collection of tools, drivers, and libraries that make the foundation of the Jetson Nano computing environment. Among the libraries, we find CUDA version 10.2. The Cuda Deep Neural Network Library provides optimized deep learning routines like convolutions, poolings, activation functions and more. Several CUDA-specific profilers and debuggers are given as well. The Jetson Nano image ships with a comprehensive general purpose Computer Vision SDK as well, including a pre-compiled version of the well-known OpenCV framework.

Each model of Jetson is either available as a development kit or as a production module. The difference is in the form of the board, smaller for production, and in the pre-installed software. The development board comes with several tweaking features, the most important being the ability to enforce different power budgets. This ability will be crucial for our evaluation.

The experiments described here have been implemented using scientific frameworks such as torch and torchvision. A note has to be made on their version, however. The software environment has been different between the training and testing phases. The latter was basically due to the intrinsic difference of the computing platforms. The inference has been carried out in an embedded scenario with limitations mainly concerning the availability of the latest versions in the system's package manager or from external sources. The algorithm was trained on a Cloud computing resource, with more powerful hardware and better support for the latest framework updates.

Power management. The power consumption of the board is given by the sum of three components: the carrier board, which comprehends the CPU and common interfaces such as the network; the hardware accelerator module, hence the Maxwell GPU; the peripheral consumption. The carrier module is rated to run between 0.5 W and 1.5 W of power with a maximum current absorption of 4 A [60].

The Jetson board comes with pre-defined power profiles. These profiles can be enabled programmatically without the need of a reboot. The main power scenario is the default one and do not limit performances, hence power consumption will be as defined by the board's instruction manual. From now on we are going to refer to this profile as *MAXN* or *nominal power setting*. A second power model limits the overall computing capacity and will be used to compare experimental results and data. Power consumption is cut by 50%, hence reduced to 5 W, by mainly switching offline two of the four available CPU cores. Moreover, the GPU clock frequency is reduced by about 40%. We refer to this profile *5* W or *minimal power setting*. As stated many times, restricting power consumption is vital when considering long-lasting deployment applications. Moreover, optimal power management is mandatory in applicative contexts where a standalone device must run for a long period of time.

More scenarios are technically supported when it comes to power profiles. The Jetson boards can be overclocked, drastically modifying the board behavior and power usage. Software procedure can bring the CPU clock up to 2 MHz and the GPU to 1 MHz [61]. The consequence is that power consumption can reach 20 W, which is the range of more advanced Jetson models. We did not go further investigating such a scenario as not relevant to our objectives.

Jetson Nano platform quirks. As mentioned above, the Jetson Nano comes with a pre-defined, ready to use, OS image that can be burnt into an SD card and directly booted. The supported OS is Ubuntu in its 18.04 version, released in 2018 and decommissioned in 2022. The current version of Python as of the time of writing is 3.11, while the Jetson ships 3.6. The implementation of algorithms and training tasks has been done using the latest version of Python, thus taking advantage of new library functions and language features. The Jetson Nano is tied to Python 3.6 because of the official Jetpack, that has been decommissioned by Nvidia. Therefore no support for new Python language versions or other software tools is given.

Interestingly enough Nvidia does not officially distribute pre-built packages of major Deep Learning frameworks such as torch or tensorflow, leaving to the end user the job of compiling the whole package. That is rather annoying and time consuming. As the Nano seems to be widely adopted by both the scientific and industrial communities, these packages can be found online as already built and ready to be installed packages [61], together with a newer image based on Ubuntu 20.[3] We therefore used this solution for performing our test. It is worth noting that the evolution of this SBC, named Jetson Orin Nano, is more powerful but also more power-hungry and costly. Nevertheless, the analysis we are going to present remains valid.

4.2. The object detection stage

The object detection results achieved with the three models are reported in Table 2. We can see that ULO and YOLOv3 present comparable detection capabilities on test data. The two architectures have respectively 61.5M and 3.9M parameters. The tiny version of ULO is displaced by 6% from its original version and by a 7% from YOLOv3. ULO Tiny however achieves good detection figures, considering the fact it reduces the total parameters of its network down to 3.4M.

We then analyzed the observed runtime performance of each model. We present results by summing the total *forward time* for each test. The forward time is defined as the time to process an input image to produce the results, i.e. the bounding boxes, their classes, and confidence values. Hence we are only considering the time spent in computing class and bounding box predictions from an input image. As discussed before, we consider the two different power models.

Nominal power setting. In Table 3 the obtained results are shown. We can see that in terms of current absorption, the models present similar performance. ULO Tiny, being the computationally lighter model, achieves a lower average forward time. That translates to a higher *Frame Per Second* (FPS) ratio.

Fig. 3 presents the comparison between ULO, ULO Tiny, and YOLOv3 in terms of forward time. We can see that ULO Tiny is the fastest and has less variability. YOLOv3 even if it is far more complex in terms of computations (49.47 GFlops), achieves faster results than ULO (3.42 GFlops).

In Fig. 4 we can see that current absorption is higher when running ULO. We also note a high count of outliers that might depend on measurement error. We also note a negative skew for all three models. This means that most values are under the median.

[2] The I2C hardware interface is used to measure power consumption.

[3] https://github.com/Qengineering/Jetson-Nano-Ubuntu-20-image.

Table 2
The obtained results in object detection, expressed by mAP with an IOU threshold of .5.

Model	mAP							Validation set mAP
	Diplodus vulgaris	*Oblada melanura*	*Chromis chromis*	*Diplodus sargus*	*Spicara maena*	*Coris julis*	Mean	Mean
ULO	0.674	0.654	0.555	0.551	0.550	0.581	0.594	0.621
ULO Tiny	0.596	0.607	0.448	0.458	0.517	0.571	0.533	0.575
YOLOv3	0.627	0.639	0.663	0.524	0.553	0.643	0.608	0.630

Table 3
Measured runtime performances with the nominal power setting of Jetson Nano.

Model	Avg. current absorption	Avg. forward time	Total time	FPS
ULO	1456 mA (±62.94)	0.23 s (±0.02)	231.09 s	4.3
ULO Tiny	1402 mA (±102.42)	0.19 s (±0.02)	186.88 s	5.2
YOLOv3	1411 mA (±99.96)	0.22 s (±0.04)	214.92 s	4.4

Table 4
Measured runtime performances with the minimal power setting of Jetson Nano.

Model	Avg. current absorption	Avg. forward time	Total time	FPS
ULO	915.28 mA (±135.91)	0.27 s (±0.05)	260.12 s	3.7
ULO Tiny	867.41 mA (±136.97)	0.24 s (±0.06)	231.12 s	4.1
YOLOv3	1102.15 mA (±244.34)	0.30 s (±0.05)	288.96 s	3.3

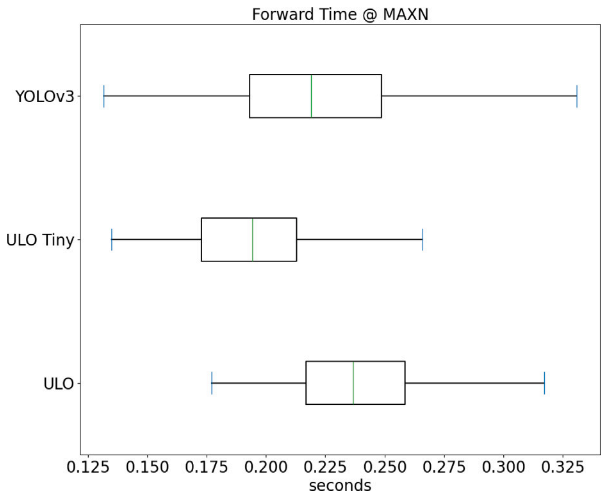

Fig. 3. The forward time distribution of the three models.

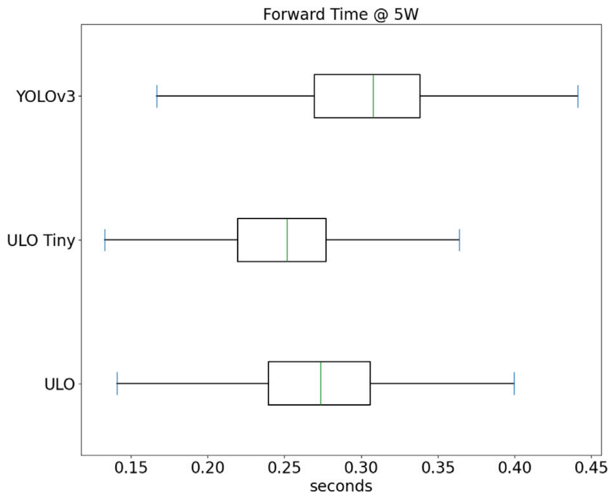

Fig. 5. The forward time distribution of the three models when running in the 5 W power profile.

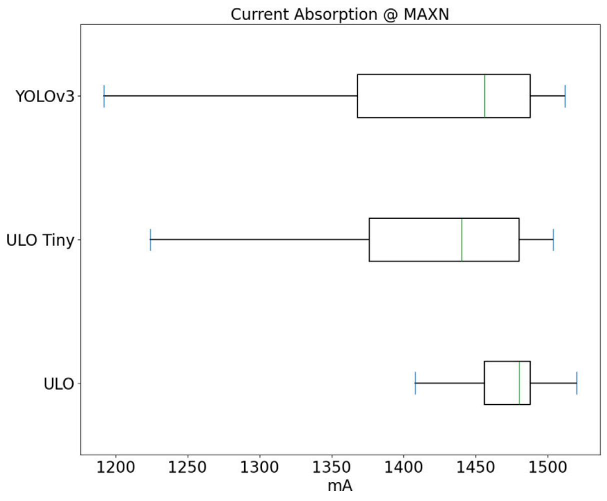

Fig. 4. An overview of the current consumption distributions for each model.

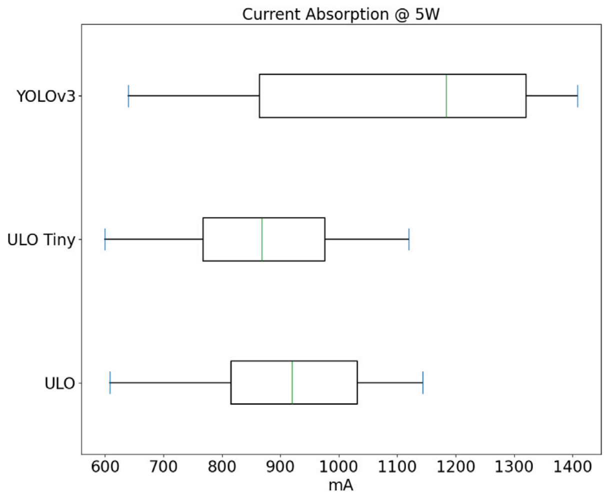

Fig. 6. The current absorption distribution of three models when running in the 5 W power profile.

Minimal power setting. We run the same experiments by switching to the 5 W power profile. Results are shown in Table 4, Figs. 5 and 6. We can see that now absorption varies slightly more than what is observed in Table 3. Moreover, YOLOv3 still absorbed on average over 1000 mA. In general, with this power setting the models loose about 1 frame every second compared to the previous scenario.

At this stage of tests, the ULO Tiny architecture has the highest turnout. We observed a significantly lower average current absorption. The time difference between ULO Tiny compared to ULO and YOLOv3 has shrunk, as also the FPS, compared to what was obtained previously. Such performance figures are as expected when running in a constrained environment.

Moreover, when computational resources are limited, we observe a distinct difference between YOLO and the two computationally smaller

Table 5
A tabular view of the profiles of ULO and ULO Tiny. Data corresponds to I/O and preparation times, st. means startup. The last column lists the average time necessary to process each image considering all the steps performed on the edge device. Results are in seconds.

Model - #Images	Data	NMS	Weights Load	Python st.	CUDA st.	Inference	Avg.per image
ULO - 1	8.43	0.25	1.66	20.71	12.38	0.23	43.66
ULO Tiny - 1	7.44	0.67	1.21	19.84	12.85	0.17	42.18
ULO - 100	14.97	11.79	1.88	20.00	13.56	15.40	0.77
ULO Tiny - 100	14.34	9.62	1.25	20.66	15.59	12.51	0.73
ULO - 500	35.68	65.11	1.47	20.69	18.30	80.03	0.44
ULO Tiny - 500	35.70	43.31	1.12	17.47	17.03	70.37	0.37
ULO - 1000	61.05	119.41	1.51	20.61	19.69	151.65	0.37
ULO Tiny - 1000	59.93	81.40	1.21	19.82	12.47	143.32	0.31

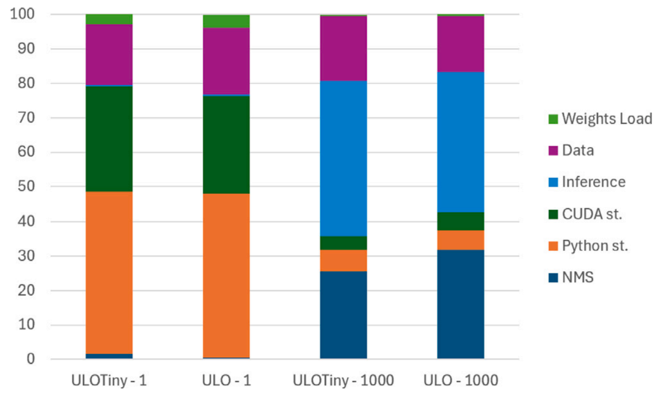

Fig. 7. The time profile of the complete ULO and ULO Tiny detection pipeline when launched for 1 and 1000 images.

models. The displacement in current consumption is clearly noticeable, suggesting that such a model would not be optimal for long-lasting deployments with respect to ULO and ULO Tiny. Comparing Figs. 4 and 6 we notice an evident difference in the consumption variance for ULO.

4.3. *The full fish detection pipeline*

Based on the previous results, we propose here a brief comparison between ULO and ULO Tiny-based pipelines in terms of runtime, considering all the steps of the fish detection pipeline. We consider the scenario of an underwater observatory with a limited power source, a fixed camera, and the possibility of varying the frequency of the image processing. In the case of AUV in fact the image analysis must be activated nearly in real time, to determine the proper course.

In particular, the Jetson Nano can be activated following two distinct policies, i.e. on every new image acquisition or with a certain time frame based on the image acquisition frequency.

The main purpose of such a scenario would be at first to optimize the data transfer between the observatory and the ground-controlling station and possibly to change the time frame on the basis of what is present in the last image.

The expected behavior, in fact, is that the edge computing platform is able to recognize valuable targets among the imagery and choose which of them has to be sent. For example, in the selected dataset described in Section 3.1, only 28% of the acquired images contain fishes belonging to the classes of interest listed in Table 1. Our goal is to evaluate a balance between the pipeline initialization time costs with the amount of time spent doing inference on imaging.

In detail, we measured the time spent by program initialization, reading and loading in memory the object detection architecture, processing required on an image before the detection stage and other overheads. We also give insights on the warmup time, and the initial elaboration latency that the GPU experience when idle. Table 5 shows the results.

We start by considering the case when each image acquired by the camera triggers the execution of the pipeline. At this stage, we can assess that there are no significant differences between the two runs. The only noticeable difference is the time spent reading the weights file of the architectures: ULO Tiny, having fewer parameters, has a smaller weights file resulting in a saving of 1 s.

The initial delay due to the Python and CUDA runtime is considerably high and dominates the processing time, as shown in Fig. 7 for both models. During the experiments, we measured such delay by using a randomly initialized image and passed it as input for the model. We also note that the GPU latency time impacts the image resize. This is the reason why, in our implementation, we decided to read the file image and transfer it immediately to the GPU memory, in order to perform the resize with the GPU.

We could balance the time spent on warm-up by increasing the number of processed images. But this consideration is related to the applicative context, as described above. For instance, if the observatory needs to respond as soon as a certain type of fish is detected (for example by taking more pictures or changing the direction in case of AUV), activating the object detection pipeline further ahead in time might not be of great interest. Instead, if the focus of the observatory is just to report to shore interesting images, letting any update in the frequency of the acquisition and processing to the Cloud component of the system, the computations can be delayed when a sufficient amount of images are acquired.

When we run the pipeline on a large image batch we can see that the processing cost is very low, in the order of half a second per image, and very close to the forward time of Table 4. The reason is clearly the fact that the setup overhead is spread among many images and now the inference time is the most important component, as shown in Fig. 7. Furthermore, there is little difference between processing 500 or 1000 images, while the time to process only 100 images remains relevant. These results show that more than 100 images are necessary to let the system reach an effective operating regime.

5. Discussion

From the experimental results reported in Tables 2–5 we can conclude that, in general, ULO Tiny represents a power- and compute-effective solution for moving the image processing on the underwater monitoring system. But it does not represent always the best choice, because we must take into considerations all the different operative conditions characterizing underwater devices and observatories.

In particular, the three key factors are represented by the power supply (i.e. battery operated or not), the available bandwidth (narrow if based on a satellite link or wide through cable or GPRS connection), and the need to process images in real time or just send periodic reports. Table 6 presents the possible scenarios and the best model to be selected.

The most demanding configuration is the first one, where the model has to be selected in order to provide the best trade-off between power consumption, execution time, and precision. The main goal in this scenario is to limit the transmission of images without relevant subjects – in our case fishes – followed by the need to classify them quickly

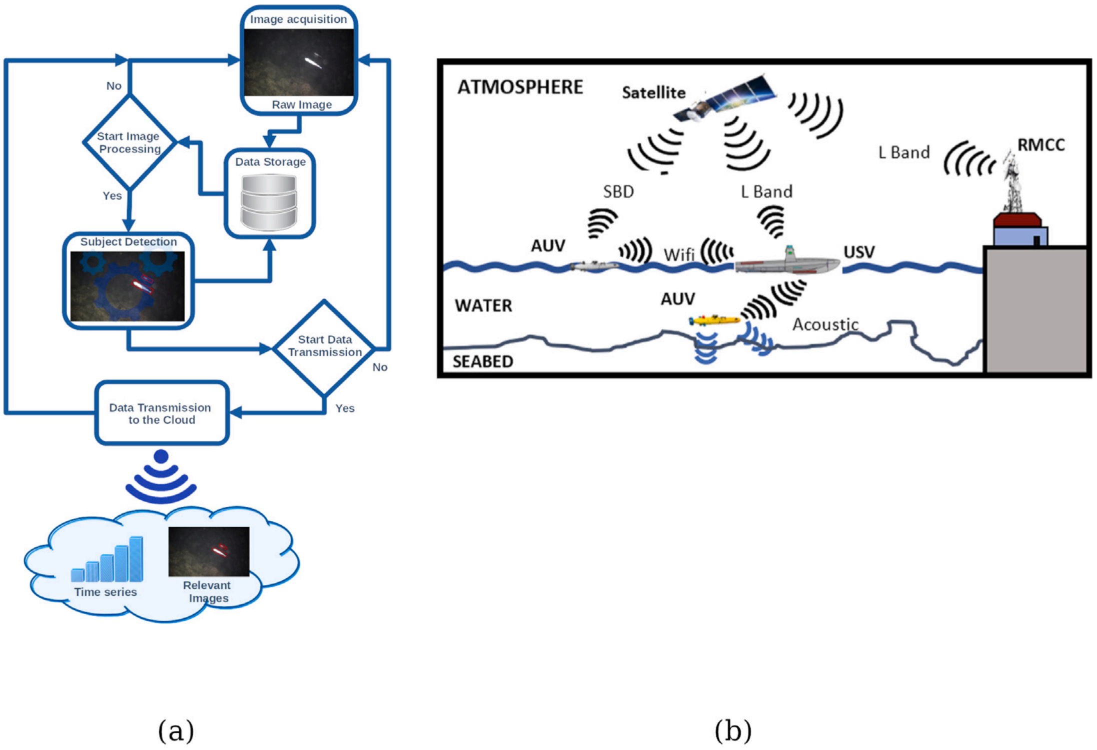

Fig. 8. The schema depicts (a) the logical functioning of the imaging device GUARD-1 equipped with onboard image processing capabilities and in (b) an example of data transfer pipeline implemented in the H2020 ENDURUNS project (Courtesy of Sanchez et al. [13]), where the image data are acquired by a GUARD-1 device installed on board of an AUV then transferred to a USV and finally transferred to a land station through satellite transmission.

Table 6
The guideline for selecting the most effective model on the basis of the different operative conditions.

Power	Transmission	Bandwidth	Model
Battery	Real time	Narrow	YOLOv3 @ 5 W
		Wide	ULO Tiny @ 5 W
	Periodic	Narrow	ULO @ 5 W
		Wide	ULO Tiny @ 5W
Wired	Real time	Narrow	YOLOv3 @ MAXN
		Wide	ULO Tiny @ MAXN
	Periodic	Narrow	YOLOv3 @ MAXN
		Wide	*Execute in Cloud*

without using too much power. YOLOv3 operated with the 5 W profile represents the best choice because it is the most accurate one — thus reducing the images to be transmitted, while its execution time and power consumption values are between the other two models.

The less demanding configuration instead is the last one: in this case the best solution could be to transfer each image directly to the Cloud, for processing them with a more complex model running on high-end servers. The other cases result in a change of relative importance among the three key factors. For example, the *battery-periodic-narrow* configuration requires to maximize the trade-off between power consumption and accuracy, because a periodic transmission does not require the fastest execution time. Instead the *battery-periodic-wide* configuration focuses mainly on the battery consumption, because a slower and less precise model is feasible if its power consumption is the minimum one.

It is worth noting that the presented analysis has a general value. These guidelines hold true for scenarios with similar characteristics, for example in environmental monitoring through images in remote locations with a limited availability of power and/or bandwidth. This is because the pipeline is based on general-purpose object detection models that, with specific training, can be used in different applicative domains.

6. From the Edge to the Cloud

A complete and detailed understanding of marine environments requires large volumes of heterogeneous data acquired continuously over extended periods of time, to capture daily, seasonal, annual, and multi-year dynamics. It is therefore clear the importance of deploying a network of spatially distributed autonomous and intelligent observatories, including underwater intelligent imaging devices, capable of acquiring heterogeneous data for extended periods in time. However, at present, they are responsible only of image acquisition, that are transferred to Unmanned Surface Vehicles (USV) via cables or acoustic devices and, when at the surface, via radio or satellite communication to a base station or directly to a Cloud infrastructure.

In recent years, many international projects have been implemented to create online archives for collecting and sharing scientific datasets [62,63], with recent implementations focusing on image data [64]. Nevertheless, most of these Cloud infrastructures provides only data download and visualizations services, without incorporating any science discovery solution based on AI approaches [65].

Moreover, not enough efforts have been devoted to harvesting the hitherto unexplored information buried in the serendipitous data collected in these repositories, as in other research fields [66]. For example, it is of particular interest the detection and tracking of alien species [67,68] because of climate change. The key reason is the lack of appropriate analysis methodologies and tools that support scientists in generating new hypotheses, designing new experiments, and providing new insights on their interpretation within the context of Science Discovery [65].

This is also due to the fact that effective data processing tools need a relevant amount of computational resources and can be executed only on high performance infrastructures [69], as the Cloud. For example, science discovery approaches need multiprocessing facilities [70–72] and more advanced approaches based on pre-trained large linguistic models cannot be executed on laboratory hardware resources [73].

Some ongoing projects aim to fill this gap by establishing the first prototypes of Virtual Labs for complex marine data processing [74]. This represents the state-of-the-art approach for Marine science but suffers the key issue of requiring a huge amount of data to be transmitted from remote observatories, even if they do not contain any relevant information. For example, in [75] the imaging device GUARD-1, installed on an autonomous drifting vehicle for deep sea observations, acquired 4020 images and only eight of them had relevant content. Since that kind of autonomous vehicles operates in the open sea, they are equipped with an Iridium modem for SBD satellite communications, capable of transmitting only small-size packages of data - i.e. 240 bytes per package [76]. Similarly, in [77] the GUARD-1 acquired underwater images every 10 min, continuously during the daylight and the night, in the period February-May 2017, for a total of 12,331 images. The images with relevant content increased as the summer period approached.

The Compute Continuum-based approach proposed in this article represents a step forward in the direction of the data-driven understanding of the marine environment. It combines AI-based intelligent observatories, Cloud services, and hardware/software solutions to reduce energy consumption and transmitted data volume [78,79].

As part of this pipeline, the GUARD-1 imaging device equipped with onboard processing capabilities could operate as described in Fig. 8(a). Depending on the application context, the acquired images can be stored onboard or processed by a classification algorithm in order to extract and save the relevant image content (e.g. number and organism species per image). After the image processing, the device can produce and transmit to the Cloud periodic reports containing the organisms' abundance time series and/or (part of) the images with relevant content. The Fig. 8(b) summarizes the communication schema implemented in the H2020 ENDURUNS project [12,80], where an AUV was equipped with the GUARD-1 camera whose images were transferred to an USV, then to a remote land station through satellite transmission.

The ENDURUNS system is the perfect example where the proposed pipeline would have improved communication performance and, consequently, AUV and USV autonomy.

Currently, an end-to-end solution including the imaging device GUARD-1 is under development within the project *Robotics and AI for Socio-economic Enpowerment* (RAISE) [81], where an observing system based on a network of heterogeneous underwater sensors distributed in a mussel-farm is going to acquire meteorological, biological, chemical and physical data together with images of the macro and mega fauna. The objective is the investigation of the ecological dynamics of harmful species impacting mussel cultivation. We plan to perform real experiments in this context.

7. Conclusions and future developments

This paper presented an Object Detection pipeline for Marine Science. Specifically, the pipeline provides an end-to-end solution based on the compute continuum paradigm from the acquisition of the underwater images, their processing using an edge computing device followed by the transmission of relevant results to a Cloud computing infrastructure.

The key features we considered in the development are the minimization of the execution and response time together with the minimization of the power consumption, as part of the pipeline will run in an underwater environment where severe power restrictions apply. Both these goals allow a reduction in the amount of transferred data using available networks.

Most of the available related works apply Deep Learning methods that require complex computations not compatible with the above scenario. Besides this, the widely used data source, Fish4Knowledge, is not suitable for the Mediterranean sea, where the operating conditions differ.

We presented performance figures using a state of the art YOLOv3 object detector with respect to ULO and its Tiny version. Results have been collected using an Nvidia Jetson Nano, being a relatively cheap option with a low power consumption that well suits our requirements.

We showed that, with the minimal power settings, the variance between the three models is high. That might be a valuable decision point if the observation mission needs to run for weeks, or even months where preferring a low power footprint might be more adequate and have a higher detection performance. This is the case of Autonomous Underwater Devices, which need to sustain long-lasting missions underwater without any human intervention. In these scenarios, any small power difference can become a valuable matter.

Then we profiled how to configure the Edge computing part of the pipeline considering the case of underwater observatories, investigating how different choices impact the total execution times. In particular, we investigated whether it is more convenient to launch the detection pipeline once every image acquisition or elaborate larger batches. Results demonstrate that devices like the Jetson Nano spend a lot of time in setup steps, like the initialization of the Python interpreter and the CUDA environment. These insights allow us to properly tune the analysis frequency considering if the image elaboration is necessary to react to immediate events or only to reduce the data that an underwater object needs to transfer to shore.

The presented analysis has a general value in every scenario with similar characteristics, for example in environmental monitoring through images when limited power and/or bandwidth are available. This is because we focused on object detection models that, with specific training, can be used in different applicative domains.

The take away message of this work is that there is not a single model that perfectly fits all the possible operative conditions. We summarized in Table 6 a guideline for selecting the most effective model, and we will exploit it for our in-situ experiments, that represent the main future development of this work. In particular, the GUARD-1 imaging device equipped with the algorithms proposed in this work will be implemented within the project RAISE [81].

Then we will focus on the further development of the Cloud component for advanced analysis of the acquired data. The functionalities of the Blue Cloud services [74] will be exploited for gathering and analyzing the images of underwater organisms together with the physical, chemical, and biological data collected within the RAISE project, for a better understanding of the behavior of Mediterranean commercial species under varying environmental conditions.

CRediT authorship contribution statement

Michele Ferrari: Writing – original draft, Validation, Software, Data curation. **Daniele D'Agostino:** Writing – review & editing, Supervision, Conceptualization. **Jacopo Aguzzi:** Writing – review & editing, Resources, Data curation, Conceptualization. **Simone Marini:** Writing – review & editing, Validation, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was partially funded by the European Union - NextGenerationEU and by the Ministry of University and Research (MUR), National Recovery and Resilience Plan (PNNR), Mission 4, Component 2, Investment 1.5, project "RAISE - Robotics and AI for Socio-economic Empowerment" - (ECS00000035); The co-author Simone Marini is part of the critical mass of the RAISE Innovation Ecosystem.

It is also funded by the Project "National Biodiversity Future Center - NBFC" funded under the National Recovery and Resilience Plan (PNNR), Mission 4 Component 2 Investment 1.4 - Call for tender No. 3138 of 16 December 2021, rectified by Decree n.3175 of 18 December 2021 of Italian Ministry of University and Research funded by the European Union – NextGenerationEU; Project code CN_00000033, Concession Decree No. 1034 of 17 June 2022 adopted by the Italian Ministry of University and Research, CUP D33C22000960007.

The research was also supported by the ICM-CSIC "Severo Ochoa Centre Excellence" (CEX2019-000928-S) and the Research Unit Tecnoterra (ICM-CSIC/UPC). Funds were also from DIGI4ECO (grant number 101112883 - GAP-101112883).

References

[1] M. Jahanbakht, W. Xiang, L. Hanzo, M.R. Azghadi, Internet of underwater things and big marine data analytics—a comprehensive survey, IEEE Commun. Surv. Tutor. 23 (2) (2021) 904–956.

[2] C.N. Bianchi, A. Azzola, S. Cocito, C. Morri, A. Oprandi, A. Peirano, S. Sgorbini, M. Montefalcone, Biodiversity monitoring in Mediterranean marine protected areas: Scientific and methodological challenges, Diversity 14 (1) (2022) 43.

[3] J. Aguzzi, D. Chatzievangelou, J.B. Company, L. Thomsen, S. Marini, F. Bonofiglio, F. Juanes, R. Rountree, A. Berry, R. Chumbinho, C. Lordan, J. Doyle, J. del Rio, J. Navarro, F.C. De Leo, N. Bahamon, J.A. García, P.R. Danovaro, M. Francescangeli, V. Lopez-Vazquez, P. Gaughan, The potential of video imagery from worldwide cabled observatory networks to provide information supporting fish-stock and biodiversity assessment, ICES J. Mar. Sci. 77 (7–8) (2020) 2396–2410, http://dx.doi.org/10.1093/icesjms/fsaa169.

[4] V. Lopez-Vazquez, J.M. Lopez-Guede, S. Marini, E. Fanelli, E. Johnsen, J. Aguzzi, Video image enhancement and machine learning pipeline for underwater animal detection and classification at cabled observatories, Sensors 20 (3) (2020) 726.

[5] S. Marini, F. Bonofiglio, L.P. Corgnati, A. Bordone, S. Schiaparelli, A. Peirano, Long-term automated visual monitoring of Antarctic benthic fauna, Methods Ecol. Evol. 13 (8) (2022) 1746–1764, http://dx.doi.org/10.1111/2041-210X.13898.

[6] S. Marini, F. Bonofiglio, L.P. Corgnati, A. Bordone, S. Schiaparelli, A. Peirano, Long-term high resolution image dataset of Antarctic Coastal Benthic Fauna, Sci. Data 9 (1) (2022) http://dx.doi.org/10.1038/s41597-022-01865-7.

[7] A. Peirano, A. Bordone, L.P. Corgnati, S. Marini, Time-lapse recording of yearly activity of the sea star Odontaster validus and the sea urchin Sterechinus neumayeri in Tethys Bay (Ross Sea, Antarctica), Antarct. Sci. 35 (1) (2023) 4–14.

[8] U. of Washington, The NEPTUNE concept: A regional cabled ocean observatory in the Northeast Pacific ocean, 2000, https://tinyurl.com/53tc9s9e. (Accessed August 2024).

[9] M. Francescangeli, S. Marini, E. Martínez, J. Del Río, D.M. Toma, M. Nogueras, J. Aguzzi, Image dataset for benchmarking automated fish detection and classification algorithms, Sci. Data 10 (1) (2023) 5.

[10] A. Benetazzo, F. Ardhuin, F. Bergamasco, L. Cavaleri, P.V. Guimarães, M. Schwendeman, M. Sclavo, J. Thomson, A. Torsello, On the shape and likelihood of oceanic rogue waves, Sci. Rep. 7 (1) (2017) http://dx.doi.org/10.1038/s41598-017-07704-9.

[11] K.J. Morris, B.J. Bett, J.M. Durden, V.A. Huvenne, R. Milligan, D.O. Jones, S. McPhail, K. Robert, D.M. Bailey, H.A. Ruhl, A new method for ecological surveying of the abyss using autonomous underwater vehicle photography, Limnol. Oceanogr.: Methods 12 (11) (2014) 795–809.

[12] S. Marini, N. Gjeci, S. Govindaraj, A. But, B. Sportich, E. Ottaviani, F.P.G. Márquez, P.J. Bernalte Sanchez, J. Pedersen, C.V. Clausen, F. Madricardo, F. Foglini, F. Bonofiglio, L. Barbieri, M. Antonini, Y.S. Montenegro Camacho, P. Weiss, K. Nowak, M. Peer, T. Gobert, A. Turetta, E. Chatzidouros, D. Lee, D. Zarras, T. Steriotis, G. Charalambopoulou, T. Yamas, M. Papaelias, ENDURUNS: An integrated and flexible approach for Seabed survey through autonomous mobile vehicles, J. Mar. Sci. Eng. 8 (9) (2020) http://dx.doi.org/10.3390/jmse8090633.

[13] P.J.B. Sanchez, F.P.G. Márquez, S. Govindara, A. But, B. Sportich, S. Marini, V. Jantara, M. Papaelias, Use of UIoT for offshore surveys through autonomous vehicles, Pol. Marit. Res. 28 (3) (2021) 175–189, http://dx.doi.org/10.2478/pomr-2021-0044.

[14] S. Marini, L. Corgnati, L. Mazzei, E. Ottaviano, B. Isoppo, S. Aliani, A. Conversi, A. Griffa, GUARD1: An autonomous system for gelatinous zooplankton image-based recognition, in: OCEANS 2015-Genova, IEEE, 2015, pp. 1–7.

[15] R. Lumpkin, M. Pazos, Measuring Surface Currents with Surface Velocity Program Drifters: the Instrument, Its Data, and Some Recent Results, Cambridge University Press, 2007, pp. 39–67, http://dx.doi.org/10.1017/CBO9780511535901.003.

[16] C.I. Addey, Using biogeochemical argo floats to understand ocean carbon and oxygen dynamics, Nat. Rev. Earth Environ. 3 (11) (2022) 739, http://dx.doi.org/10.1038/s43017-022-00341-5.

[17] F. Bonofiglio, F.C. De Leo, C. Yee, D. Chatzievangelou, J. Aguzzi, S. Marini, Machine learning applied to big data from marine cabled observatories: A case study of sablefish monitoring in the NE Pacific, Front. Mar. Sci. 9 (2022) 842946.

[18] L. Bacchiani, G. De Palma, L. Sciullo, M. Bravetti, M. Di Felice, M. Gabbrielli, G. Zavattaro, R. Della Penna, Low-latency anomaly detection on the edge-cloud continuum for industry 4.0 applications: the SEAWALL case study, IEEE Internet Things Mag. 5 (3) (2022) 32–37, http://dx.doi.org/10.1109/IOTM.001.2200120.

[19] L. Sun, X. Jiang, H. Ren, Y. Guo, Edge-cloud computing and artificial intelligence in internet of medical things: Architecture, technology and application, IEEE Access 8 (2020) 101079–101092, http://dx.doi.org/10.1109/ACCESS.2020.2997831.

[20] D. D'Agostino, L. Morganti, E. Corni, D. Cesini, I. Merelli, Combining edge and cloud computing for low-power, cost-effective metagenomics analysis, Future Gener. Comput. Syst. 90 (2019) 79–85, http://dx.doi.org/10.1016/j.future.2018.07.036.

[21] Y. Kalyani, R. Collier, A systematic survey on the role of cloud, fog, and edge computing combination in smart agriculture, Sensors 21 (17) (2021) http://dx.doi.org/10.3390/s21175922.

[22] H. Wu, Z. Zhang, C. Guan, K. Wolter, M. Xu, Collaborate edge and cloud computing with distributed deep learning for smart city internet of things, IEEE Internet Things J. 7 (9) (2020) 8099–8110, http://dx.doi.org/10.1109/JIOT.2020.2996784.

[23] EuCloudEdgeIoT.eu - the European cloud edge IoT continuum for business and research, 2022, URL https://eucloudedgeiot.eu/wp-content/uploads/2022/12/Joint-Press-Release-with-multipliers-_December-2022-1.pdf.

[24] S. Moreschini, F. Pecorelli, X. Li, S. Naz, D. Hästbacka, D. Taibi, Cloud continuum: The definition, IEEE Access 10 (2022) 131876–131886, http://dx.doi.org/10.1109/ACCESS.2022.3229185.

[25] L. Kong, J. Tan, J. Huang, G. Chen, S. Wang, X. Jin, P. Zeng, M. Khan, S.K. Das, Edge-computing-driven internet of things: A survey, ACM Comput. Surv. 55 (8) (2022) 1–41.

[26] C. Savaglio, G. Fortino, M. Zhou, J. Ma, Device-Edge-Cloud Continuum: Paradigms, Architectures and Applications, Springer Nature, 2023.

[27] A. Mijuskovic, A. Chiumento, R. Bemthuis, A. Aldea, P. Havinga, Resource management techniques for cloud/fog and edge computing: An evaluation framework and classification, Sensors 21 (5) (2021) http://dx.doi.org/10.3390/s21051832.

[28] M. Garofalo, G. Morabito, M. Fazio, A. Celesti, M. Villari, Workflow engines in the compute continuum: a comparative analysis, in: Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing, UCC '23, Association for Computing Machinery, New York, NY, USA, 2024, http://dx.doi.org/10.1145/3603166.3632148.

[29] Mid4CC '23: Proceedings of the 1st International Workshop on Middleware for the Computing Continuum, Association for Computing Machinery, New York, NY, USA, 2023.

[30] E. Fazeldehkordi, T.-M. Grø nli, A survey of security architectures for edge computing-based IoT, IoT 3 (3) (2022) 332–365.

[31] D. Rosendo, A. Costan, P. Valduriez, G. Antoniu, Distributed intelligence on the edge-to-cloud continuum: A systematic literature review, J. Parallel Distrib. Comput. 166 (2022) 71–94, http://dx.doi.org/10.1016/j.jpdc.2022.04.004.

[32] M. Aldinucci, R. Birke, A. Brogi, E. Carlini, M. Coppola, M. Danelutto, P. Dazzi, L. Ferrucci, S. Forti, H. Kavalionak, G. Mencagli, M. Mordacchini, M. Pasin, F. Paganelli, M. Torquati, A proposal for a continuum-aware programming model: From workflows to services autonomously interacting in the compute continuum, in: 2023 IEEE 47th Annual Computers, Software, and Applications Conference, COMPSAC, 2023, pp. 1852–1857, http://dx.doi.org/10.1109/COMPSAC57700.2023.00287.

[33] A. Costantini, G. Di Modica, J.C. Ahouangonou, D.C. Duma, B. Martelli, M. Galletti, M. Antonacci, D. Nehls, P. Bellavista, C. Delamarre, D. Cesini, IoTwins: Toward implementation of distributed digital twins in industry 4.0 settings, Computers 11 (5) (2022) http://dx.doi.org/10.3390/computers11050067.

[34] M. Jansen, A. Al-Dulaimy, A.V. Papadopoulos, A. Trivedi, A. Iosup, The SPEC-RG reference architecture for the compute continuum, in: 2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid), 2023, pp. 469–484, http://dx.doi.org/10.1109/CCGrid57682.2023.00051.

[35] A. Saleh, M. Sheaves, M.R. Azghadi, Computer vision and deep learning for fish classification in underwater habitats: A survey, Fish Fish. 23 (4) (2022) 977–999, http://dx.doi.org/10.1111/faf.12666.

[36] A. Rova, G. Mori, L.M. Dill, One fish, two fish, butterfish, trumpeter: Recognizing fish in underwater video, in: IAPR Conference on Machine Vision Applications, 2007.

[37] C. Spampinato, D. Giordano, R. Di Salvo, Y.-H.J. Chen-Burger, R.B. Fisher, G. Nadarajan, Automatic fish classification for underwater species behavior understanding, in: Proceedings of the First ACM International Workshop on Analysis and Retrieval of Tracked Events and Motion in Imagery Streams, ACM, 2010, http://dx.doi.org/10.1145/1877868.1877881.

[38] G. Cutter, K. Stierhoff, J. Zeng, Automated detection of rockfish in unconstrained underwater videos using haar cascades and a new image dataset: Labeled fishes in the wild, in: 2015 IEEE Winter Applications and Computer Vision Workshops, IEEE, 2015, pp. 57–62.

[39] T.U. of Edinburgh, Fish4Knowledge project, 2023, https://homepages.inf.ed.ac.uk/rbf/Fish4Knowledge/. (Accessed August 2024).

[40] A. Salman, A. Jalal, F. Shafait, A. Mian, M. Shortis, J. Seager, E. Harvey, Fish species classification in unconstrained underwater environments based on deep learning, Limnol. Oceanogr.: Methods 14 (9) (2016) 570–585.

[41] H. Qin, X. Li, J. Liang, Y. Peng, C. Zhang, DeepFish: Accurate underwater live fish recognition with a deep architecture, Neurocomputing 187 (2016) 49–58.

[42] X. Li, M. Shang, H. Qin, L. Chen, Fast accurate fish detection and recognition of underwater images with fast R-CNN, in: OCEANS 2015 - MTS/IEEE Washington, IEEE, 2015, http://dx.doi.org/10.23919/oceans.2015.7404464.

[43] A. Salman, S.A. Siddiqui, F. Shafait, A. Mian, M.R. Shortis, K. Khurshid, A. Ulges, U. Schwanecke, Automatic fish detection in underwater videos by a deep neural network-based hybrid motion learning system, ICES J. Mar. Sci. 77 (4) (2020) 1295–1307.

[44] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.

[45] A. Al Muksit, F. Hasan, M.F.H.B. Emon, M.R. Haque, A.R. Anwary, S. Shatabda, YOLO-fish: A robust fish detection model to detect fish in realistic underwater environment, Ecol. Inform. 72 (2022) 101847.

[46] L. Chen, M. Zheng, S. Duan, W. Luo, L. Yao, Underwater target recognition based on improved YOLOv4 neural network, Electronics 10 (14) (2021) 1634.

[47] D. Heller, M. Rizk, R. Douguet, A. Baghdadi, J.-P. Diguet, Marine objects detection using deep learning on embedded edge devices, in: 2022 IEEE International Workshop on Rapid System Prototyping, RSP, IEEE, 2022, pp. 1–7.

[48] M. Paraschiv, R. Padrino, P. Casari, E. Bigal, A. Scheinin, D. Tchernov, A. Fernández Anta, Classification of underwater fish images and videos via very small convolutional neural networks, J. Mar. Sci. Eng. 10 (6) (2022) 736.

[49] L. Wang, X. Ye, S. Wang, P. Li, ULO: An underwater light-weight object detector for edge computing, Machines 10 (8) (2022) 629.

[50] S. Marini, E. Fanelli, V. Sbragaglia, E. Azzurro, J. Del Rio Fernandez, J. Aguzzi, Tracking fish abundance by underwater image recognition, Sci. Rep. 8 (1) (2018) 13748.

[51] U.P. de Catalunya Barcelonatech, et al., Expandable seafloor observatory, 2023, https://obsea.es/. (Accessed August 2024).

[52] J. Redmon, A. Farhadi, YOLO9000: better, faster, stronger, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7263–7271.

[53] A. Chen, A. Chow, A. Davidson, A. DCunha, A. Ghodsi, S.A. Hong, A. Konwinski, C. Mewald, S. Murching, T. Nykodym, et al., Developments in mlflow: A system to accelerate the machine learning lifecycle, in: Proceedings of the Fourth International Workshop on Data Management for End-To-End Machine Learning, 2020, pp. 1–4.

[54] R. Bonghi, Jetsonstats, 2023, https://github.com/rbonghi/jetson_stats. (Accessed August 2024).

[55] S. Marini, F. Bonofiglio, L.P. Corgnati, A. Bordone, S. Schiaparelli, A. Peirano, Long-term automated visual monitoring of Antarctic benthic fauna, Methods Ecol. Evol. 13 (8) (2022) 1746–1764.

[56] S. Marini, A. Griffa, S. Aliani, A. Conversi, K. Schroeder, M. Borghini, Ep2863257 (a1) - underwater images acquisition and processing system, 2013, https://data.epo.org/gpi/EP2863257B1.

[57] J. Aguzzi, D. Chatzievangelou, S. Marini, E. Fanelli, R. Danovaro, S. Flögel, N. Lebris, F. Juanes, F.C. De Leo, J. Del Rio, L. Thomsen, C. Costa, G. Riccobene, C. Tamburini, D. Lefevre, C. Gojak, P.-M. Poulain, P. Favali, A. Griffa, A. Purser, D. Cline, D. Edgington, J. Navarro, S. Stefanni, S. D'Hondt, I.G. Priede, R. Rountree, J.B. Company, New high-tech flexible networks for the monitoring of deep-sea ecosystems, Environ. Sci. Technol. 53 (12) (2019) 6616–6631.

[58] J. Aguzzi, D. Chatzievangelou, M. Francescangeli, S. Marini, F. Bonofiglio, J. del Rio, R. Danovaro, The hierarchic treatment of marine ecological information from spatial networks of benthic platforms, Sensors 20 (6) (2020) http://dx.doi.org/10.3390/s20061751.

[59] L. Verderame, I. Merelli, L. Morganti, E. Corni, D. Cesini, D. D'Agostino, A. Merlo, A secure cloud-edges computing architecture for metagenomics analysis, Future Gener. Comput. Syst. 111 (2020) 919–930, http://dx.doi.org/10.1016/j.future.2019.09.013.

[60] Nvidia, Jetson nano user guide, 2023, https://developer.nvidia.com/embedded/downloads. (Accessed August 2024).

[61] QEngineering, Qengineering, 2023, https://qengineering.eu/. (Accessed: August 2024).

[62] L. Pecci, M. Fichaut, D. Schaap, SeaDataNet, an Enhanced Ocean Data Infrastructure Giving Services to Scientists and Society, Vol. 509, 2020, http://dx.doi.org/10.1088/1755-1315/509/1/012042.

[63] J. Felden, L. Möller, U. Schindler, R. Huber, S. Schumacher, R. Koppe, M. Diepenbroek, F.O. Glöckner, PANGAEA - data publisher for earth & environmental science, Sci. Data 10 (1) (2023) 347, http://dx.doi.org/10.1038/s41597-023-02269-x.

[64] K. Katija, E. Orenstein, B. Schlining, L. Lundsten, K. Barnard, G. Sainz, O. Boulais, M. Cromwell, E. Butler, B. Woodward, K.L.C. Bell, FathomNet: A global image database for enabling artificial intelligence in the ocean, Sci. Rep. 12 (1) (2022) 15914, http://dx.doi.org/10.1038/s41598-022-19939-2.

[65] H. Wang, T. Fu, Y. Du, W. Gao, K. Huang, Z. Liu, P. Chandak, S. Liu, P. Van Katwyk, A. Deac, A. Anandkumar, K. Bergen, C.P. Gomes, S. Ho, P. Kohli, J. Lasenby, J. Leskovec, T.-Y. Liu, A. Manrai, D. Marks, B. Ramsundar, L. Song, J. Sun, J. Tang, P. Veličković, M. Welling, L. Zhang, C.W. Coley, Y. Bengio, M. Zitnik, Scientific discovery in the age of artificial intelligence, Nature 620 (7972) (2023) 47–60, http://dx.doi.org/10.1038/s41586-023-06221-2.

[66] A. Belfiore, S. Carpano, D. D'Agostino, F. Haberl, D. Law-Green, G. Lisini, et al., The extras project: Exploring the X-ray transient and variable sky, Astron. Astrophys. 650 (2021) A167.

[67] S. Giakoumi, S. Katsanevakis, P.G. Albano, E. Azzurro, A.C. Cardoso, E. Cebrian, A. Deidun, D. Edelist, P. Francour, C. Jimenez, V. Mačić, A. Occhipinti-Ambrogi, G. Rilov, Y.R. Sghaier, Management priorities for marine invasive species, Sci. Total Environ. 688 (2019) 976–982, http://dx.doi.org/10.1016/j.scitotenv.2019.06.282.

[68] A. Zenetos, M. Galanidi, Mediterranean non indigenous species at the start of the 2020s: recent changes, Mar. Biodivers. Rec. 13 (1) (2020) 10, http://dx.doi.org/10.1186/s41200-020-00191-4.

[69] L. Guidi, A. Fernandez Guerra, C. Canchaya, E. Curry, F. Foglini, J.-O. Irisson, K. Malde, C.T. Marshall, M. Obst, R.P. Ribeiro, J. Tjiputra, D.C. Bakker, Big Data in Marine Science, Tech. Rep., European Marine Board, Ostend, Belgium, ISBN: 9789492043931, 2020, http://dx.doi.org/10.5281/zenodo.3755793.

[70] D.J. Papworth, S. Marini, A. Conversi, A novel, unbiased analysis approach for investigating population dynamics: A case study on Calanus finmarchicus and its decline in the North Sea, PLOS ONE 11 (7) (2016) 1–26, http://dx.doi.org/10.1371/journal.pone.0158230.

[71] M. Landajuela, C.S. Lee, J. Yang, R. Glatt, C.P. Santiago, I. Aravena, T. Mundhenk, G. Mulcahy, B.K. Petersen, A unified framework for deep symbolic regression, in: S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, A. Oh (Eds.), Advances in Neural Information Processing Systems, Vol. 35, Curran Associates, Inc., 2022, pp. 33985–33998.

[72] N. Makke, S. Chawla, Interpretable scientific discovery with symbolic regression: a review, Artif. Intell. Rev. 57 (1) (2024) 2, http://dx.doi.org/10.1007/s10462-023-10622-0.

[73] B. Romera-Paredes, M. Barekatain, A. Novikov, M. Balog, M.P. Kumar, E. Dupont, F.J.R. Ruiz, J.S. Ellenberg, P. Wang, O. Fawzi, P. Kohli, A. Fawzi, Mathematical discoveries from program search with large language models, Nature 625 (7995) (2024) 468–475, http://dx.doi.org/10.1038/s41586-023-06924-6.

[74] Blue cloud - a federated European FAIR and open research ecosystem for oceans, seas, coastal and inland waters, 2024, https://blue-cloud.d4science.org. (Accessed August 2024).

[75] S. Marini, L. Corgnati, L. Mazzei, E. Ottaviano, B. Isoppo, S. Aliani, A. Conversi, A. Griffa, GUARD1: An autonomous system for gelatinous zooplankton image-based recognition, in: OCEANS 2015 - Genova, 2015, pp. 1–7, http://dx.doi.org/10.1109/OCEANS-Genova.2015.7271704.

[76] Iridium short burst data (SBD), 2024, https://www.iridium.com/services/iridium-sbd/. (Accessed August 2024).

[77] S. Marini, L. Corgnati, C. Mantovani, M. Bastianini, E. Ottaviani, E. Fanelli, J. Aguzzi, A. Griffa, P.-M. Poulain, Automated estimate of fish abundance through the autonomous imaging device GUARD1, Measurement 126 (2018) 72–75, http://dx.doi.org/10.1016/j.measurement.2018.05.035.

[78] D. D'Agostino, A. Quarati, A. Clematis, L. Morganti, E. Corni, V. Giansanti, D. Cesini, I. Merelli, SoC-based computing infrastructures for scientific applications and commercial services: Performance and economic evaluations, Future Gener. Comput. Syst. 96 (2019) 11–22.

[79] D. D'Agostino, I. Merelli, M. Aldinucci, D. Cesini, Hardware and software solutions for energy-efficient computing in scientific programming, Sci. Program. 2021 (2021) 1–9.

[80] P.J.B. Sanchez, F.P.G. Marquez, M. Papaelias, S. Marini, S. Govindaraj, Enduruns project: Advancements for a sustainable offshore survey system using autonomous marine vehicles, in: J. Xu, F. Altiparmak, M.H.A. Hassan, F.P. García Márquez, A. Hajiyev (Eds.), Proceedings of the Sixteenth International Conference on Management Science and Engineering Management – Volume 1, Springer International Publishing, Cham, 2022, pp. 363–378.

[81] RAISE - robotics and AI for socio-economic enpowerment, 2024, https://www.raiseliguria.it/. (Accessed August 2024).

Michele Ferrari received his MsC in Computer Science in 2023. Since 2019 he is employed at FlairBit, a company that develops IoT and data analysis solutions. He mainly works on mobile and Bluetooth development to support raw data collection from hardware devices.

Daniele D'Agostino, Ph.D., is associate professor at the Department of Computer Science, Bioengineering, Robotics and Systems Engineering of the University of Genova. His research activities concern the design of science gateways in different research fields, and the development of parallel software for e-Science. He co-organized the 22th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, several special issues on ISI journals and co-authored more than 100 scientific papers, published in journals, book chapters and conference proceedings.

Jacopo Aguzzi research focuses on changes in sampled richness and biodiversity at different temporal scales, extracting ecological indicators from imaging products by cabled observatories, crawlers, ROVs, AUVs and neutrino telescopes. He is manager for the "ecological monitoring" and "citizen science" of the EMSO Testing-Site OBSEA (www.obsea.es). Between 2019-2019, he has been Scientific Advisor for the Ocean Network Canada (ONC) for the Section "Life in the Environments of the Northeast Pacific Ocean and Salish Sea".

Simone Marini Ph.D. is researcher at the Institute of Marine Sciences of the National Research Council of Italy. His research activity deals with knowledge discovery and pattern analysis, study and design of intelligent marine observing systems, where he is first inventor of the European patent EP2863257. He leaded several research units and workpackages within national and international projects, he co-authored more than 100 scientific publications among which 70 with impact factor. He edited several special issues and he is associated editor of two ISI scientific journals. He acted as reviewer for different research programs, included the European Research council.