



Integration of object detection and semantic segmentation based on convolutional neural networks for navigation and monitoring of cyanobacterial blooms in lentic water scenes

Fredy Barrientos-Espillo^{a,*}, María J. Gómez-Silva^a, Eva Besada-Portas^a, Gonzalo Pajares^b

^a Department of Computer Architecture and Automation, University Complutense of Madrid, Madrid 28040, Spain

^b Institute for Knowledge Technology, University Complutense of Madrid, Madrid 28040, Spain

HIGHLIGHTS

- Dual-task architecture based on CNNs for simultaneous detection of objects and amorphous semantic structures.
- SPP facilitates the detection of objects of different sizes and textures, while PAN merges information from diverse scales.
- CBAM improves the detection of small objects of interest in regions affected by adverse environmental conditions.
- Lentic water datasets annotated using Grounding DINO and Segment Anything Model (SAM), and then manually refined.
- Train and evaluate model configurations with the same backbone, neck, and detection head, but varying the segmentation head.

ARTICLE INFO

Keywords:

Dual-task
Object detection
Semantic segmentation
Convolutional Neural Networks
Autonomous surface vehicles
Lentic waters
Cyanobacterial blooms

ABSTRACT

Lentic waters, such as lakes, lagoons, reservoirs, and wetlands are characterized by their absence of current. In recent decades, they have been threatened by pollution and scarcity due to various environmental factors. Therefore, they require frequent monitoring to ensure their health and purity, especially to control the proliferation of harmful cyanobacteria (pollutants). Machine Vision Systems (MVS) on board Autonomous Surface Vehicles (ASVs) is a good option for automatic image processing in this context. ASVs must navigate safely, and obstacle detection is essential. In addition, the segmentation of pollutants in water is crucial. We propose an architecture based on convolutional neural networks that integrates both object detection and semantic segmentation. The goal is to simultaneously extract all available global information to detect objects and amorphous textures (cyanobacterial patches and water bodies), considering their variations in size, pose, and appearance. The architecture includes two branches: object detection and semantic segmentation, sharing the same backbone and neck. We evaluate the model on our dataset and the results show that it can holistically understand lentic water scenes with high accuracy, and the integration of the attention mechanism improves its overall performance.

1. Introduction

Lentic waters (lakes, lagoons, reservoirs, and wetlands) are those freshwater bodies that remain still, not flowing in one direction. In recent decades, they have been threatened by pollution and scarcity due to different environmental factors. Of particular interest is the sporadic and unforeseeable occurrence of cyanobacterial harmful blooms (pollutants). All of this makes frequent monitoring necessary to ensure their

health and purity. Given this situation, nowadays, there are systems capable of automatically taking measurements at strategic points and alerting the competent authorities about contamination events [1]. However, these early warning systems cannot perform dynamic monitoring of pollutants.

On the other hand, Machine Vision Systems (MVS) onboard Autonomous Surface Vehicles (ASVs) [2] are a good option for automatic water quality monitoring as they can collect spatio-temporal dynamic

* Corresponding author.

E-mail addresses: fredybar@ucm.es (F. Barrientos-Espillo), mgomez77@ucm.es (M.J. Gómez-Silva), ebesada@ucm.es (E. Besada-Portas), pajares@ucm.es (G. Pajares).

<https://doi.org/10.1016/j.asoc.2024.111849>

Received 16 October 2023; Received in revised form 27 April 2024; Accepted 29 May 2024

Available online 15 June 2024

1568-4946/© 2024 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

information by capturing images and do not require permanent human intervention with a high degree of autonomy. ASVs have the advantage that they can carry other onboard sensory systems in addition to the MSV for both pollutant detection and navigation support, e.g., to measure the level of nitrogen concentration, pH, GPS location, and others. In addition, MSVs offer the possibility that they can also monitor the water surface when installed statically on the shore.

ASVs must navigate safely on the water surface and obstacle detection is essential in this regard with a complete understanding of their environment from the MVS point of view. But more importantly, the detection of pollutants in a water body is of great interest, both under the paradigm of texture identification and localization based on the images captured by the MVS. All this while operating in environments with several visual understanding challenges: a broad variety of landscapes (dynamic background), sunlight reflections, reflections of objects, objects, vegetation or buildings on the water surface, different types and a number of objects, and different textures (water, pollutants, grasses, trees, sky, etc.). Therefore, two tasks must be addressed simultaneously: object detection and texture identification.

Object detection in MSV identifies and locates elements present at the scene, typical of these environments, including canoes, people on board, or paddles for navigation, submerged rocks, and floating tree trunks, among others. These elements can damage or interfere with the navigation of ASVs if not properly detected and avoided. On the other hand, semantic segmentation provides an overall understanding of the scene by determining the various amorphous textures present, such as water, cyanobacterial blooms, grasses, trees, sky, and others. As mentioned above, the goal is to simultaneously detect objects and textures (with special emphasis on cyanobacteria patches) considering that both, objects and textures, show different sizes, poses, and appearances on the images under the adverse environmental conditions outlined above.

Object detection [3–6] and semantic segmentation [7–11] approaches based on deep learning are relatively mature and the requirements for data annotation are straightforward, which in the early years were studied separately. In recent years, different studies in the field of autonomous driving [12–19] proved that the integration of object detection and semantic segmentation in a unique network architecture, sharing different elements and components, achieves better performance over single detection and segmentation [14,15,17]. Some of these techniques, such as the one proposed in [12] identify objects by means of the corresponding bounding boxes and segment their content at the pixel level, i.e., semantic segmentation. On the other hand, one of the objectives for addressing any computer vision problem, such as ours, is to determine the most favorable network architecture to increase the receptive field to identify objects and semantic structures outside of objects, unlike in [12]. In object detection methods, based on Convolutional Neural Networks (CNNs), this results in selecting the best possible structures such as backbone, neck, and head, which are the key components of this type of model. In semantic segmentation, the relevant structures are: a) the encoder, for extracting high-level features from the incoming image to capture spatial information at different scales, by reducing the spatial dimensions of feature maps while increasing their number of channels; b) the decoder, that takes the feature maps from the encoder and up-samples (including deconvolutions) them to the size of the original image, with the aim of reconstructing fine-grained details while reducing the number of channels in the feature maps; c) skip connections, linking feature maps from the encoder to the corresponding feature maps in the decoder, allowing the decoder to access high-resolution features existing in the decoder, helping to preserve the spatial details during the up-sampling process and improving segmentation accuracy.

Under these considerations, we exploit the well-known efficiency and performance of YOLOv4 [20] and YOLOv5 object-based detectors to carry out the detection process, while semantic segmentation takes advantage of backbone and neck modules in YOLOv4/YOLOv5, that act

as encoder and then we use a head, that acts as decoder involving up-sampling operations, for pixel labeling.

In addition, specific modules are introduced in both, the object detector, and the semantic segmentation, to emphasize and enhance the information flowing through them for detecting objects and textures at different sizes and exploiting both global and local information in the images through the feature maps in the network. Local information is of particular relevance in the case of cyanobacterial textures, as they display amorphous structures and no homogeneity of intensity levels in the spectral channels of the images. The CNN-model design for simultaneous object detection and semantic segmentation contains three main parts: backbone, neck, and two heads. The backbone and neck are consecutive modules feeding both heads. One head is specific for detecting objects and the second head is for identifying textures. Each one of these parts is based mostly on the incorporation of specific modules to extract the required simultaneous information in the envisaged scenarios, valid for both, object detection and texture identification. The basis for object detection is YOLOv3 [44] and the basis for texture identification are three well-tested approaches in semantic segmentation (BiSeNet [38], DeepLabV3+ [39], PSPNet [10]). All these parts are conveniently integrated, sharing information, and in each of them are integrated specific modules oriented towards the objective of identifying objects and textures simultaneously. This integrated and shared design makes the main contribution of this paper, where the selection of specific modules embedded in each part is decisive.

Other than this, to our knowledge, there are no combined methods of sharing information for image understanding (detection of objects and textures) in lentic water. The common practice is to apply each approach separately, as sketched in Table 1. This table contains a summary of references in this regard, displaying the structure of the network models (encoder-decoder), the application scenarios, the type of elements detected, along with the limitations of each model, and the improvements proposed in our approach.

In general, these studies share the use of CNNs approaches for the detection and segmentation of elements in aquatic environments, but separately, which is an important difference with respect to our proposal. Moreover, they differ in the specific elements sought to be detected, the segmentation methods used, and the type of images or scenes used as input to the network, but all seek to improve the ability of ASVs to navigate and operate in aquatic environments.

The approach in [21] focuses on maritime obstacle detection based on image segmentation. The aim is to detect small objects while addressing the issue of reflections on the water surface. It is based on a U-Net [8] encoder-decoder architecture. The encoder, based on the Xception structure [22] with separable convolutions, extracts deep features, while the decoder fuses, through Feature Fusion Modules (FFM) [38], high-level features (containing global information with positions of objects and shapes) and low-level features (local characteristics of objects, such as edges, corners, and salient textures). Separable convolutions are usually not appropriate for capturing spatial relationships, due to the nature of such convolutions. This approach uses the Attention Refinement Module (ARM) [38] focusing on modeling the interdependencies between channels in feature maps and capturing global information, but it does not consider local information useful for texture description. This clearly represents a limitation since in our scenarios with amorphous cyanobacteria textures spatial relations are critical. Furthermore, the focus on the identification of small objects is another handicap in scenarios where object sizes are highly variable.

Similarly, the study [23] focuses on the segmentation and refinement of maritime obstacle detection. The model contains an image encoder based on ResNet101 [24] with atrous convolutions producing feature maps. The encoder also contains feature channels with pixel values set according to the horizon location based on an inertial measurement unit (IMU). The horizon projection into the image allows to constrain water-edge estimation. The decoder fuses the visual and inertial information from these feature maps and increases the spatial resolution of

Table 1

Studies that use semantic segmentation algorithms based on Deep learning for the detection of elements in aquatic environments.

Ref.	Encoder	Decoder	Scenario	Detected elements	Limitations	Improvements
[21]	Derived from Xception network.	Feature Fusion Modules (FFM) to concatenate feature maps at different dimensions.	Maritime	Sea, sky, and marine obstacles (buoys, floating objects, and ships).	• Uses exclusively an image segmentation approach. • Only small objects are detected. • Separable convolutions have difficulty capturing spatial relationships in textures. • ARM (Attention Refinement Module) with global average pooling, deals only with global aspects, without considering local information, mainly on textures.	• Share a backbone and neck, feeding both a well-proven object detector and three semantic segmentation modules. • Detection of objects with different sizes. • CBAM to capture global and contextual (blooms) information.
[23]	ResNet101 with atrous convolutions) and an Inertial Measurement Unit (IMU).	Fuses the visual and inertial information from the encoders and increases the spatial resolution of the segmentation output.	Maritime	Marine objects (buoys, floating objects, and ships), water areas, and water edges.	• Only objects are of interest to be detected. • Assumes the majority of pixels in marine scenes belong to either water or sky. • IMU feature maps help to delimit the water edge to separate water and sky. It must be synchronized. • ARM with global average pooling • Only objects are of interest to be detected.	• Object and different textural structures including blooms. • Only images are required, no IMU. • CBAM attention mechanism to capture global and contextual (blooms) information.
[25]	The encoder part of ENet, ESPNet, and SegNet.	The decoder part of ENet, ESPNet, and SegNet.	Maritime	Sea, sky, and marine objects.	• ENet and ESPNet focus on computational efficiency. • Only water surfaces are to be detected. • No specific mechanisms (SPP, CBAM) are added to capture objects and textures at different resolutions. • No object detection mechanisms are considered.	• Object and different textural structures. • Fast and accurate dual-task network for real-time object and texture detection. • Objects and textures simultaneously are to be detected. • Additional SPP and CBAM mechanisms can capture both local and global contextual information.
[26]	Convolutional layers with 32, 64, 128, 256, and 512 filters, respectively.	FCN adapted for water segmentation.	Lake	Water areas	• Only water surfaces are to be detected. • No specific mechanisms (SPP, CBAM) are added to capture objects and textures at different resolutions. • No object detection mechanisms are considered.	• Objects and textures simultaneously are to be detected. • Additional SPP and CBAM mechanisms can capture both local and global contextual information.
[27]	The encoder part of FCN, SegNet, and U-Net	The decoder part of FCN, SegNet, and U-Net.	Lake	Water areas	• Only water surfaces are to be detected with semantic segmentation. • No object detection mechanisms are considered. • As before, no specific mechanisms SPP and CBAM are considered.	• Objects and textures simultaneously are to be detected. • Additional SPP and CBAM mechanisms can capture both local and global contextual information.
[28]	Ten 3×3 convolutional layers, and by four 2×2 max pooling operations with stride 2.	Eight 3×3 convolutional layers and four 2 × 2 transpose layers.	Lake	Water areas and waterline.	• Only the water surface and waterline are detected. • No object detection mechanisms are considered. • Uses a limited dataset (191 labeled images).	• Objects and textures simultaneously are to be detected. • A dataset representative of the diversity of lentic aquatic scenarios.
[29]	The encoder part of U-Net.	The decoder part of U-Net.	Lake	Water areas	• Only water surfaces are to be detected. • No object detection is considered. • The training labels depend on the lidar information.	• Object and different textural structures including blooms. • Data Annotation with Grounding DINO [63] and Segment Anything Model (SAM) [64].
[30]	The encoder part of SegNet.	The decoder part of SegNet.	River	Water areas	• Only water surfaces are to be detected. • No object detection mechanisms are of interest. • The images used for training correspond to a specific river.	• Object and different textural structures including blooms. • A dataset representative of the diversity of lentic aquatic scenarios.
[31]	ResNet50 (FCN), ResNet101 (DeepLabV2)	UperNet (FCN), Atrous spatial pyramid pooling (DeepLabV2).	River	Water areas and river level.	• Only water areas and river levels are detected. • No object detection is considered. • It does not provide details on the diversity of the images used for training.	• Objects and textures simultaneously are to be detected. • The collected images exhibit a wide range of features.
[32]	A residual block is added to each convolution block.	Up-sampling layers and skip connections.	Remote sensing	Water areas	• Only water surfaces are to be detected. • No object detection is considered.	• Objects and textures simultaneously are to be detected.

(continued on next page)

Table 1 (continued)

Ref.	Encoder	Decoder	Scenario	Detected elements	Limitations	Improvements
[33]	ResNet50 and EfficientNet-b6	UNet++, FPN, PSPNet, and DeepLabV3+.	Lakes and reservoirs	Water areas and cyanobacterial blooms.	- Resnet Blocks do not capture global spatial relations between pixels - Only water areas and cyanobacterial blooms, with specific mechanisms for capturing local and global texture identification. - No object detection is considered. - The network structure used does not work in real-time.	- Additional SPP and CBAM mechanisms can capture both local and global contextual information. - Objects and textures simultaneously are to be detected. - Fast and accurate dual-task network for real-time object and texture detection.

the segmentation output. This approach also applies ARM to capture global context to refine features during the fusion in the decoder. While the study [25] focuses on the evaluation and comparison of different semantic segmentation networks in maritime surveillance environments. However, the network models proposed in these studies are adapted to work in specific maritime environments, specifically in coastal surveillance areas, which is not our case, where objects and textures are the structures of interest.

On the other hand, studies [26–29] focus on water segmentation in images containing lake scenes acquired by ASVs for use in autonomous navigation tasks. In particular, [26] proposes a Deep learning-based network of encoder-decoder structure, where the encoder is customized, and the decoder is derived from Fully Convolutional Network (FCN) [7] to detect water areas in high-resolution images. No specific mechanisms are provided to capture local and global information at different resolution levels, i.e., no different paths are designed containing Spatial Pyramid Pooling (SPP) [35] and Convolutional Block Attention Module (CBAM) [37].

In contrast, the study [27] uses FCN, SegNet [9] and U-Net [8] networks to evaluate the performance of these models, where only water and no water classes are identified. They are classical image segmentation models, with the mechanisms expressed above. The study [28] proposes a convolutional neural network architecture to detect water areas and waterlines. Meanwhile, study [29] uses U-Net network to segment water. However, these studies propose convolutional neural network architectures adapted to segment water areas, since they used a dataset, whose images are only labeled in two classes (water and non-water), which differs from our approach where objects and textures are the structures of interest. Similarly, studies [30,31] use semantic segmentation networks based on Deep learning adapted to detect water areas in images whose scenes come from rivers. These works do not consider specific mechanisms for the detection of complex textures, including blooms with different resolutions or object detection.

The approach in [32] proposes to detect water areas in remote sensing images based on semantic segmentation. However, these studies only focus on the detection of water areas, without considering other elements present in environments of this type, including objects and textures on the water surface. Moreover, the encoder uses residual blocks by directly connecting layers of the same resolution in the decoder. Residual blocks consist of two or more convolutional layers with skip connections bypassing the convolutional layers. They are designed to capture local information by comparing the input (through the bypass) and the convolved output. They have a clear disadvantage as may not capture correctly global contextual information and spatial relationships between pixels without specific additional mechanisms such as SPP or CBAM.

Finally, the work in [33] is the closest to the purposes of this work, as it focuses on the detection of cyanobacterial blooms from the ASV perspective using semantic segmentation methods based on Deep Learning with synthetic images in lake and reservoir environments. However, as before, no specific mechanisms are considered to capture

local and global contextual information against the variety of textures displayed by the blooms. It also does not consider the detection of objects for safe navigation purposes.

In general, all these studies seek to improve the ability of ASVs to navigate and segment aquatic environments using semantic segmentation techniques based on Deep learning, where scene types vary between maritime, lake, and river environments, as well as satellite images in some cases. The features detected also vary from work to work, where the segmentation of water areas and pollutants are the most common features addressed, but without specific mechanisms (SPP, CBAM) to capture both, local and global texture and object detection for safe navigation. Therefore, these studies do not perform a complete understanding of the scene, as they rely mainly on a specific task, i.e., semantic segmentation where different textures are detected but without taking into account the special characteristics of the scenarios envisaged in our approach, where the existence of blooms and their textural characteristics require the consideration of specific mechanisms to capture both local and spatial information within the more generic aquatic environments. Also, they do not consider in depth object detection and in different approaches only with global object detection but cannot identify the individual element in the crowd (e.g., a single canoe in the group of canoes, a single person in the crowd). Therefore, a complete understanding of lentic water scenes using MVS onboard ASVs or static on the shore requires an approach for simultaneous object detection and semantic segmentation, which is our proposal.

Under the above considerations, we propose a combined CNN-based model for simultaneous object detection and semantic segmentation in lentic water scenarios, to identify and localize objects (navigational obstacles) and textures, including amorphous structures such as cyanobacterial blooms, as shown in Fig. 1. The combined approach takes advantage of the benefits of object detector and semantic segmentation, improving the overall performance. The model adopts the architecture of an object detector with three parts: backbone, neck, and head. The output of the backbone is the input to the neck and the output of the neck feeds the head. The head is broken down into two branches, one for object detection purposes and the other for semantic segmentation. The backbone uses the CSPDarknet-53 [34] model as a feature extractor. To enrich the information fed into the two branches, the neck uses Spatial Pyramid Pooling (SPP) [35] to remove the fixed size constraint of the input image, the Path Aggregation Network (PAN) [36] to enhance the information flow, and the Convolutional Block Attention Module (CBAM) [37] to pay attention to the most important regions of the image, since the challenge of ASVs in lentic waters is the detection of objects in highly variable environments due to unfavorable environmental conditions, such as illumination changes, sunlight reflections, object reflections, vegetation reflections, rain and the same dynamics of the water surface, as expressed above. The two branches of the head are responsible for processing the enriched spatial information coming from the neck. The first branch, inspired by the YOLOv3 head, used in YOLOv4 and YOLOv5, performs the detection (classification and regression) of bounding boxes. On the other hand, the second semantic



Fig. 1. Samples of the integration of object detection and semantic segmentation. The images on the left side are the RGB images, and the images on the right side are the results of the integration of object detection and semantic segmentation.

segmentation branch employs 3 types of adapted segmentation heads (BiSeNet [38], DeepLabV3+ [39], PSPNet [10]) to predict the category of each pixel. The architectural details of the proposed model are explained in Section 3. For training and validation, only labeled data for object detection and semantic segmentation are needed.

The main contribution's breakdown is summarized as follows: 1) An algorithm that integrates object detection and semantic segmentation is proposed for a complete understanding of lentic water scenes. 2) The SPP mechanism is used to facilitate the detection of objects of different sizes as well as textures of different areas, and the attention mechanism called CBAM is used to identify the most critical regions of the image where environmental conditions are unfavorable. 3) The training and evaluation of the model is performed on a dataset consisting of images of lentic water scenes, which have been annotated using Grounding DINO [63] and Segment Anything Model (SAM) [64], and then manually refined. 4) Experimental results show that the model can holistically understand the lentic water scenes accurately.

The organization of this paper is as follows: Section 2 reviews related work, concerning object detection and semantic segmentation. Section 3 presents the details of the proposed model. Section 4 discusses the results of the experiment, and Section 5 draws the main conclusions obtained.

2. Related work

2.1. Object detection

Object detection aims to classify and locate objects in an image using bounding boxes. With the development of deep learning, object detection methods are basically divided into two-stage methods and one-stage methods. Among the most representative two-stage methods are R-CNN [40] and its variants Fast R-CNN [5], Faster R-CNN [41], and Mask R-CNN [42]. These methods first propose a set of candidate objects and then classify these candidates, leading to unsatisfactory overall

efficiency. Moreover, the training process is slow as it involves multiple stages and cannot be trained end-to-end. Therefore, single-stage methods were proposed, which do not have a candidate proposal module and rely on end-to-end trainable deep learning using CNNs for direct detection. Examples include RetinaNet [6], which introduces focal loss to address the class imbalance, SSD [4], which employs deconvolution to incorporate contextual information and improve detection accuracy, and YOLO [3] and its variants YOLOv2 [43], YOLOv3 [44], YOLOv4 [20] and YOLOv5, which achieve a better balance between speed and accuracy. YOLOv5 is an improved version based on YOLOv4 [45], whose main contribution is the translation of the Darknet framework to the PyTorch framework, resulting in improved performance that outperforms other models. These one-stage methods outperform two-stage methods mainly due to their training and inference speed, which makes them suitable for real-time applications. In this proposal, we take advantage of the structure of the YOLOv4 and YOLOv5 detectors, since they operate in real-time on a conventional GPU, and training only requires a conventional GPU as well. These methods have proven their efficiency in object detection, including individual objects within larger groupings. However, these methods cannot fully understand the scene, as they can only detect elements using bounding boxes, and they are not applicable to shapeless elements such as pollutants, water areas, grasses, trees, sky, which appear in our scenarios and specifically cyanobacterial harmful blooms with local and global special textural properties.

2.2. Semantic segmentation

Semantic segmentation methods, which aim to predict the category of each pixel in an image, play a crucial role in understanding complex scenarios, and completing the task of object detectors. The introduction of FCN [7], a fully convolutional network with end-to-end trainable convolutional layers, was the first approach to semantic segmentation. However, this network suffers from the problem of low resolution

caused by multiple downsamplings. As a result, several derivative and variant neural networks have been developed to address this limitation, including SegNet [9], UNet [8], PSPNet [10], and DeepLab [11]. As mentioned before, these methods are specifically designed to capture information at the pixel level and require extra effort to extract local and global contextual information, which is the problem addressed in this approach. Instance segmentation methods [42] offer a solution to this problem by combining object detection and semantic segmentation techniques, allowing individual objects to be identified from groups. However, these instance segmentation methods focus primarily on objects in the foreground of the scene, often ignoring objects in the background. In addition, these types of methods are not suitable for detecting amorphous structures such as water, as they add bounding boxes in addition to generating segmentation masks.

2.3. Integration of object detection and semantic segmentation

To address the problem of understanding the aforementioned complex scenes in the images, we propose a simultaneous object detection and image segmentation CNN-based module as it improves learning efficiency, and also allows the use of the relationship between network modules to make the model robust and applicable [14]. The multi-task network called MultiNet [46] performs joint classification, detection, and semantic segmentation through a unified network where the encoder is shared among the three tasks. The approach in [16] proposes a joint object detection and semantic segmentation algorithm, where the semantic segmentation result is preprocessed under the entropy principle, and the bounding box obtained with the object detector is optimized with K-means. Meanwhile, the method in [14] proposes an integrated object detection and semantic segmentation method with the cross and internal attention mechanisms.

The approach described in [15] also proposes a joint object detection and semantic segmentation algorithm. This algorithm consists of three modules: the encoder-decoder module, the object detection module, and the semantic segmentation module. In the encoder, they use the last four blocks of ResNet50 along with three new residual layers. To improve the semantic information of the feature maps coming from the encoder, they adopt the skip connection mechanism. In addition, they add the Squeeze-and-Excitation (SE) [49] module to the skip connection module to improve the representation capabilities of the feature maps from the decoder. In the detection module, they design a PriorBox initialization mechanism to propose more object candidates with various aspect ratios at different feature maps. The segmentation module is based on a multi-scale atrous convolution (MSAC) structure to explore global and local semantic information.

DSPNet [17] is another joint object detection and semantic segmentation method based on a ResNet50 encoder for feature extraction and two decoder branches for object detection and semantic segmentation. Both decoders receive multilevel feature maps from the encoder through a residual block, which extracts feature information. The detection branch is based on SSD [4], where multiple aspect ratios are applied to each feature layer to predict class labels and bounding boxes. The segmentation branch is inspired by PSPNet.

However, these studies propose a multi-tasking algorithm for traffic scenes, and these algorithms cannot be directly applied to lentic water scenes, where important challenges appear. Indeed, as expressed above, for object detection the complication arises due to illumination changes, sunlight reflections, object reflections, vegetation reflections, rain and water surface dynamics, and lack of labeled data, where objects of different sizes appear. On the other hand, from the point of view of semantic segmentation, the complication comes from the variability of texture properties, particularly in cyanobacterial blooms, with amorphous shapes in lentic water scenes.

In summary, in this paper, we address an inherent problem to extract as much global information as possible from the image derived from outdoor environments where the objective is to simultaneously identify

objects in the navigation environment and textures produced by cyanobacterial patches, in both cases of different dimensions and sizes. This motivates the design of the proposed network model, consisting of the modules described in the next section. Each module serves a purpose whose main reason for being is the need to all available global information by identifying objects and textures at different scales and levels of resolution, integrating both spatial and contextual information.

3. The method

Fig. 2 shows the overview of the CNN-based model architecture, which consists of three parts: a) Backbone, b) Neck, and c) Head. This last with two branches, one for object detection and one for semantic segmentation. The incoming images are processed by the Backbone, which generates feature maps at different resolution levels. These feature maps feed the Neck, where several important modules are responsible for: a) extracting relevant information at different scales (SPP module), to facilitate the detection of objects of different sizes as well as textures of different areas; b) exploiting the inter-channel and inter-spatial relationship of features (CBAM module), focusing the attention on relevant properties (enhanced) from the objects and from cyanobacteria patches, with the aim of capturing spatial contextual information at each scale and c) fusing the information from all enhanced feature maps (PAN module) at the different scales and resolutions, coming from objects and textures of different sizes. The Neck, through the PAN module, provides aggregated feature maps to the Head at three resolution levels (1/8, 1/16, 1/32). The Head consists of two parallel branches, namely: a) Object Detection and b) Semantic Segmentation. The object detection branch follows the same scheme as YOLOv4 and v5 and merges the information from the three aggregated and reinforced feature maps with the aim of predicting bounding boxes and classes. The semantic segmentation branch implements three specific semantic networks (BiSeNet, DeepLabV3+, PSPNet) with the aim of comparing their performances on this design. Each of these networks is fed with the same three aggregated and reinforced feature maps, which are enlarged and concatenated to obtain the semantic feature map. Finally, the detected objects and the semantic map are combined to output labeled objects and semantic labels at the pixel level. Table 2 summarizes the functionality of each module integrated in the proposed architecture design while identifying the purpose of each one. In the following sections, when each of the modules and sub-modules is explained, it is detailed how these functionalities are achieved.

3.1. Backbone

The backbone is based on Cross Stage Partial Network (CSPNet) [34], an efficient architecture that reduces the number of parameters and operations without losing accuracy. In addition, CSPNet addresses the problem of duplication of gradient information within network optimization. Specifically, we are inspired by the CSPDarknet-53 backbone [34] used in YOLOv4 and YOLOv5. The reason to choose this backbone is that together with the models SPP (Spatial Pyramid Pooling) [35] and PAN (Path Aggregation Network) [36], both in the Neck, it becomes the optimal backbone according to the different experiments conducted and reported in [20]. CSPDarknet-53 contains 53 (3×3) convolutional layers, a 725×725 receptive field, and 27.6 M parameters, making it more powerful than Darknet-19 [43] and more efficient than other backbones, such as CSPResNet-50 [34] and EfficientNet-B3 [47]. The backbone is responsible for extracting and combining features from the input image, its cross-stage partial connections, through the initial and deep layers, enable to extract respectively the low-level (local specifications, such as edges, corners, and textures) and high-level (global specifications, such as relations and shapes) information required for object detection and segmentation. The partial connections facilitate the propagation and enhancement of information across different stages and allow the fusion of multi-scale features. The presence of low-level

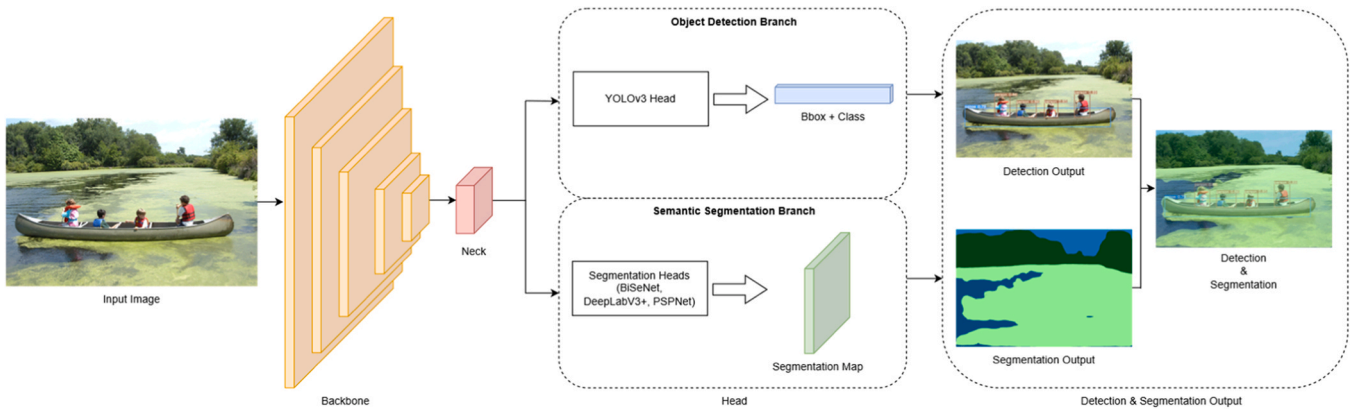


Fig. 2. Architecture overview of our dual-task model integrating object detection and semantic segmentation.

Table 2

Motivation of the model architecture, functionalities, and objectives.

	Functionality	Objective
Backbone	Generation of feature maps at different levels of resolution.	Embed the underlying information of the images.
SPP	Extract relevant information at different scales.	Facilitate the object detection and textures with different sizes and areas.
Neck	CBAM	Exploit inter-channel and inter-spatial relationship of features.
PAN	Combine the information from enhanced feature maps.	Prepare the information of objects and textures of different sizes.
Head	YOLOv3	Object detection
BiSeNet	Semantic segmentation	Locate and identify textures
DeepLabV3+ PSPNet		

features at different (including higher) stages preserves grained details while capturing semantic context. Hence its potential as a backbone for capturing, maintaining, and propagating features for object detection while preserving and enhancing semantic context, as expected according to the proposed objectives. Fig. 3 shows the backbone architecture of the proposed model. The FOCUS layer, Fig. 3, is incorporated to reduce the computational cost of the first CONV layer using a tensor reshaping strategy. Using this technique, we transform the input $x(b,c,w,h)$ into a $y(b,4c,w/2,h/2)$ representation, managing to halve the spatial resolution ($w/2$ and $h/2$) while increasing the depth by a factor of 4 ($4c$).

3.2. Neck

The neck concatenates the backbone and the head, Fig. 2. Specifically, feature maps at different resolutions from three different layers of the backbone are combined to enrich the information feeding the head, Fig. 4. The incoming feature maps come from the three respective layers whose dimensions have been previously reduced in the backbone in the proportions indicated ($1/8$, $1/16$ and $1/32$), where $1/8$ is the highest resolution and $1/32$ is the lowest one. The dimensions of the neck feature maps are of the same resolutions as the inputs. Two significant modules can be distinguished: SPP and PAN.

3.2.1. SPP

SPP module receives its input from the low-resolution feature map ($1/32$) in the backbone. SPP is really a concept that supports different implementations. Its mission is to capture/preserve spatial information at different levels of resolution to identify structures (objects, cyanobacteria patches) of different sizes. It was initially conceived to organize

the output of a feature map while maintaining spatial (hence its name) information along the sizes of the feature map [35] to provide an output towards a fully connected layer. In the proposed scheme, Fig. 4, the SPP module is designed to produce as output a feature map of a fixed dimension ($1/32$) to be subsequently upsampled and concatenated with the input of dimension $1/16$. Because these dimensions are fixed, and we need to capture spatial information at different levels of resolution, four pooling operations are applied, which are sufficient in relation to the dimensions of the input feature map. This is the approach applied in our design, inspired in [20], where a pyramidal pooling approach is applied by sliding four kernels of different sizes $k = \{1,5,9,13\}$ on every channel and applying max pooling with a stride of one. So, if the feature map contains C channels of sizes $m \times n$, each kernel produces a stack of C pooled results; in this case, four stacks are concatenated, generating the output of the SPP module. As pointed out by [35] the coarsest pyramid level ($k = 1$) performs a “global pooling” operation, keeping the same information of the channel where it is applied. In addition to preserving the spatial information, mentioned above, the pyramidal pooling operation has the great advantage of capturing image structures of different sizes, thanks to the dimensions of the filters and under different poses. This is the reason why in our work we have selected this structure, as the objects to be detected have different sizes, and more importantly, cyanobacterial patches cannot only appear but can do so with different dimensions and amorphous structures, so they can be seen under any appearance. Moreover, thanks to more-grained filters, e.g., $k = 13$, texture information can be captured from cyanobacterial patches. Experiments with different kernel dimensions do not provide any improvement to the final results. The results also do not improve with the introduction of more SPP modules in the $1/8$ and $1/16$ inputs, so it is proposed to use them exclusively in the lowest resolution input ($1/32$).

3.2.2. PAN

The concatenated feature maps coming from the SPP module are fused with the high-resolution feature maps (coming from inputs $1/16$ and $1/8$) in the PAN. The fusion is carried out after applying up-sampling/down-sampling operations with convolutions to bottom-up and top-down paths with interconnections between feature maps at the same levels of resolution with concatenation, as in [20]. PAN is a concept with the aim to enhance, propagate, and aggregate the information contained in the feature maps through different levels of resolution under a pyramidal structure. The design of this concept was originally described in [36]. It is worth noting that the PAN structure displayed in Fig. 4, differs from the PAN in [36] in the fact that we only consider two paths (bottom-up and top-down) against the three paths involved in [36], i.e., bottom-up, top-down and again bottom-up where the aggregation is applied. In our approach, the information enhancement is mainly achieved with the addition of the CBAM modules, which together with the pyramidal structure, achieve an improvement in the

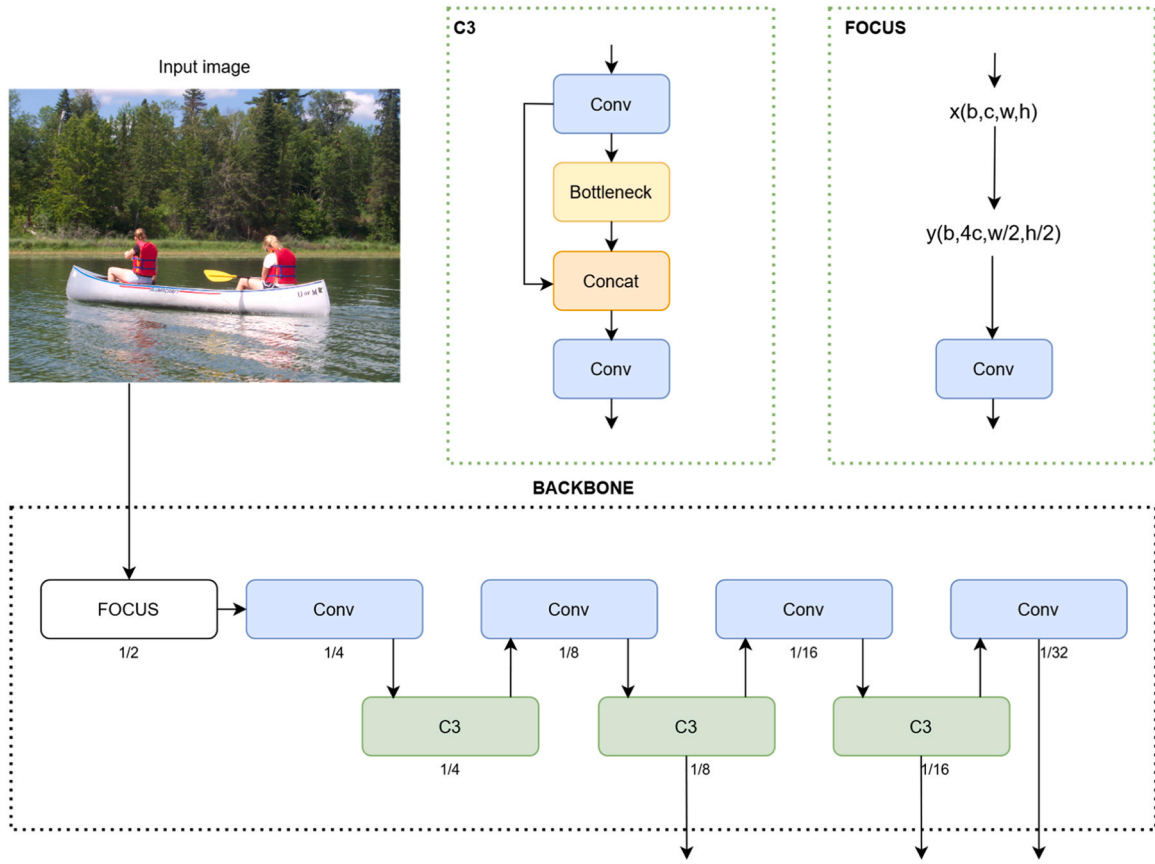


Fig. 3. Overview of the backbone architecture.

extraction of information, i.e., objects and textures at different scales. This is the main reason why we have decided to use only two paths, as it also has fewer parameters and less risk of producing unnecessary overfitting in a model that is complex in nature. The use of two paths at different resolutions is aligned with other proposals such as Feature Pyramid Network (FPN) [48], with proven performances.

The CBAM [37], outlined in Fig. 5, are structures operating at three different resolution levels, Fig. 4. They are of special relevance, whose task is to enhance the information contained in the feature maps to focus on receptive fields by applying a well-known attention mechanism in Computer Vision while assuming the functionality of the aggregation process more accurately in our scenarios. Indeed, as mentioned before, it should not be forgotten that MVS onboard ASVs, can be affected by different situations on the water surface, such as reflections on water of objects and vegetation, with different levels (low/high) of illumination or shadows, among others. Therefore, the proposed network architecture needs to effectively handle these scenarios by paying special attention to such regions. For example, it must highlight significant features under areas with low/high illumination, mainly the textures generated by the cyanobacterial patches. The inclusion of CBAM aims to improve the overall performance of the model, particularly in detecting objects more accurately within regions affected by adverse environmental conditions.

Moreover, in terms of performance, it has been shown to outperform the exclusive use of channel attention, as observed in the experiments conducted in [37], compared to the Squeeze-and-Excitation (SE) [49] module. CBAM consists of two components, namely the Channel Attention Module (CAM) and the Spatial Attention Module (SAM), which contribute to obtaining meaningful feature maps in two dimensions [37]: channel and spatial. CAM focuses on “what” is important given an input image since each channel of a feature map is considered as the feature detector [50], recovering global information. Meanwhile,

SAM focuses on “where” is an informative part [37,51], dealing with local information. In general, the CBAM block learns what and where to focus on to obtain important features and suppress unnecessary ones.

To obtain final refined features containing both global and local features, attention maps are sequentially computed in two dimensions: channel and spatial. CAM decomposes the input feature map into two vectors of dimensionality ($C \times 1 \times 1$), where C is the number of channels in the incoming feature map. One vector is generated through Global Average Pooling, while the other is generated through Global Max Pooling. Average pooling is used to aggregate spatial information, while max pooling preserves contextual information and gathers information about distinctive object features. Specifically, average pooling learns the positions and shapes of the objects of interest, while max pooling preserves object edges, corners, and textures. Both vectors are passed through the shared network, which consists of a Multilayer Perceptron (MLP) with a hidden layer. We combine the output vectors using element-wise summation to obtain channel attention.

SAM decomposes the input feature map of dimensionality ($C \times h \times w$) into 2 channels, i.e., ($2 \times h \times w$), where each channel represents max pooling and average pooling. We then concatenate them and apply a convolutional layer, resulting in a single-channel feature map with dimensions ($1 \times h \times w$), representing the spatial attention map. Finally, these attention maps are multiplied by the input feature map, as shown in Fig. 5.

CBAM modules with their respective CAM and SAM blocks are perfectly validated and contrasted to capture both spatial and contextual information, so any other design adding more blocks, does not add any improvement in the results, besides that in our proposal the CBAM blocks are used in the different resolution levels indicated in PAN, so we have decided to keep the original structure of the CBAM. On the other hand, with respect to the pooling operations applied, we have tried other strategies so that the two operations indicated in CAM and SAM are

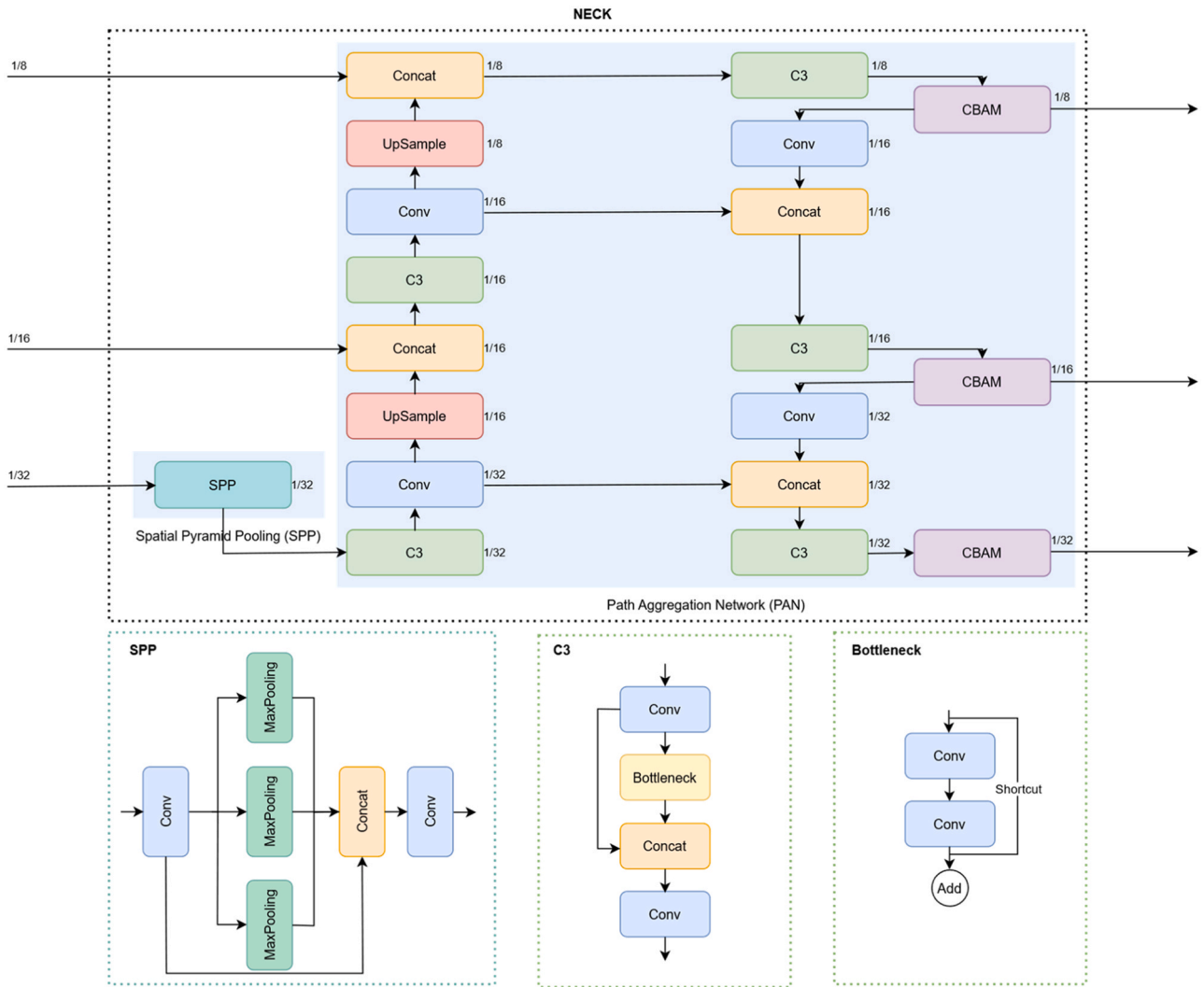


Fig. 4. Overview of the neck architecture.

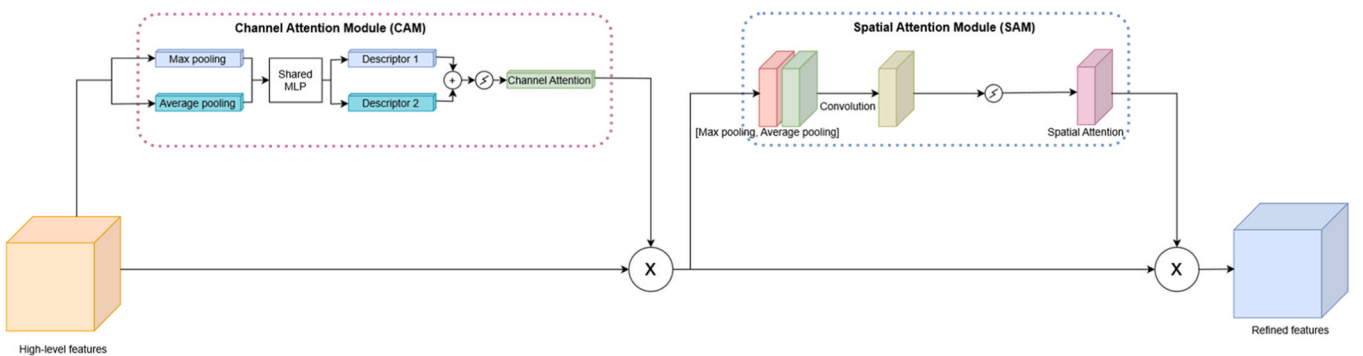


Fig. 5. Overview of the Convolutional Block Attention Module (CBAM) architecture, adapted from [37].

eliminated and replaced by a single one that integrates max and average, as they must be preserved because of their advantages, namely [52]: a) mixed, assigning weights to average and max; b) rank-based, assigning larger weights to higher activations in the pooling region. This strategy has not been successful, so we have decided to keep the CBAM module as defined in the original article.

In short, thanks to the sequence of operations in both paths of PAN, including the CBAM modules, the information in the incoming image embedded in structures of different characteristics flows at different

levels of resolution (scales) both in terms of size and nature of the textures (with special emphasis on cyanobacteria patches), which is highly beneficial in our approach.

3.3. Head: object detection branch

The object detection branch is in charge of predicting the classes of detected objects and the coordinates of bounding boxes. In this study, we exploit the structure of the YOLOv3 head, which is also used in YOLOv4

and YOLO5. This structure is anchor-based and generates three different feature maps for three spatial scales, following a configuration similar to the Feature Pyramid Network (FPN) [48]. Specifically, the input for the object detection branch is obtained from the output of the PAN block (1/8, 1/16, 1/32), as shown in Fig. 4. In prediction, we apply the DIOU [53] Non-Maximum Suppression (NMS) method which efficiently selects a single bounding box out of many overlapping bounding boxes, i.e. the one with maximum probability.

On the other hand, the bounding box regression loss function uses CIOU [53] instead of the mean square error loss, as it makes the bounding box regression faster and more accurate [20]. Meanwhile, the confidence and classification loss function use the cross-entropy loss.

3.4. Head: Semantic segmentation branch

The semantic segmentation branch takes the information from the neck (PAN) to then perform up-sampling and generate a segmentation mask as a label map. For comparative purposes and performance, we chose three heads based on the following semantic segmentation networks: BiSeNet [38], DeepLabV3+ [39], and PSPNet [10], Fig. 6. In each of these modules the head refers exclusively to the part of the module where feature maps are expanded until the final resolution level is achieved, i.e., where each pixel is assigned a class label indicating the semantic category it belongs to. The head can be assimilated to the decoding part in classical semantic segmentation structures, while the encoding part existing in semantic segmentation is done in the backbone and neck modules.

It is well-known that semantic segmentation approaches require both spatial information, including contextual information, and receptive fields of varying dimensions to consider the different sizes of structures (objects and textures). These objectives have been sufficiently addressed in previous architectures, mainly in the neck, where different arrangements and combinations of modules and operations aim to extract, enhance, and transfer precisely the spatial and contextual information at different levels of resolution to detect structures of different sizes.

The pixel-wise cross entropy is the loss function used in the three

approaches for training.

BiSeNet: is a modified version of the original paper, where two paths appear (spatial and context paths) instead of the three used in our proposal to cope with the three inputs. As mentioned before, in our architecture the spatial and contextual information have been sufficiently considered in both the backbone and the neck. Moreover, the original approach contains the Attention Refinement Module (ARM), which is used to refine the features at each stage. However, in our approach, we do not consider the ARM, as we apply the CBAM attention mechanism in PAN, producing the inputs for semantic segmentation at the three resolution levels of the feature maps (1/8, 1/16, and 1/32) feeding BiSeNet, DeepLabV3+, and PSPNet, Fig. 6. To reduce computational complexity and mimic the refinement of the original BiSeNet header, we applied a 3×3 convolutional layer followed by up-sampling by a factor of 2. In addition, we use the Feature Fusion Module (FFM) [38] to fuse the results from three inputs, resolutions 1/32 and 1/16 are previously up-sampled to 1/8 and the three concatenated inside FFM to produce a feature map, on which a convolution, batch normalization and relu are consecutively applied to obtain a new feature map M, as described in the original work [38]. Then, two flows (M1 and M2) are generated from M. M1 is obtained from M by applying global pooling and two 1×1 convolution operations followed by a sigmoid operation that multiplies the values of M to obtain M3 (an enhanced version of M). The second flow M2 is just M. M2 and M3 are finally added to produce the output of FFM. Again, this module reinforces the feature maps with the aim of capturing the required contextual information and receptive fields.

DeepLabV3+: we use 1/8 and 1/16 feature maps from the PAN block as inputs, as can be seen in the middle part of Fig. 6. These inputs were chosen based on the original DeepLabV3+ design [39], which suggests that shallow feature maps with fewer channels perform better for training. We applied a 1×1 convolutional layer and a 3×3 convolutional layer to the 1/8 input for such reduction. For the 1/16 input, we apply another 1×1 convolution followed by Atrous Spatial Pyramid Pooling (ASPP) [54] with different rates and up-sampling by a factor of 2. Then these feature maps are fused with the corresponding low-level (1/8) features using FFM. After fusion, we added a 3×3 convolutional

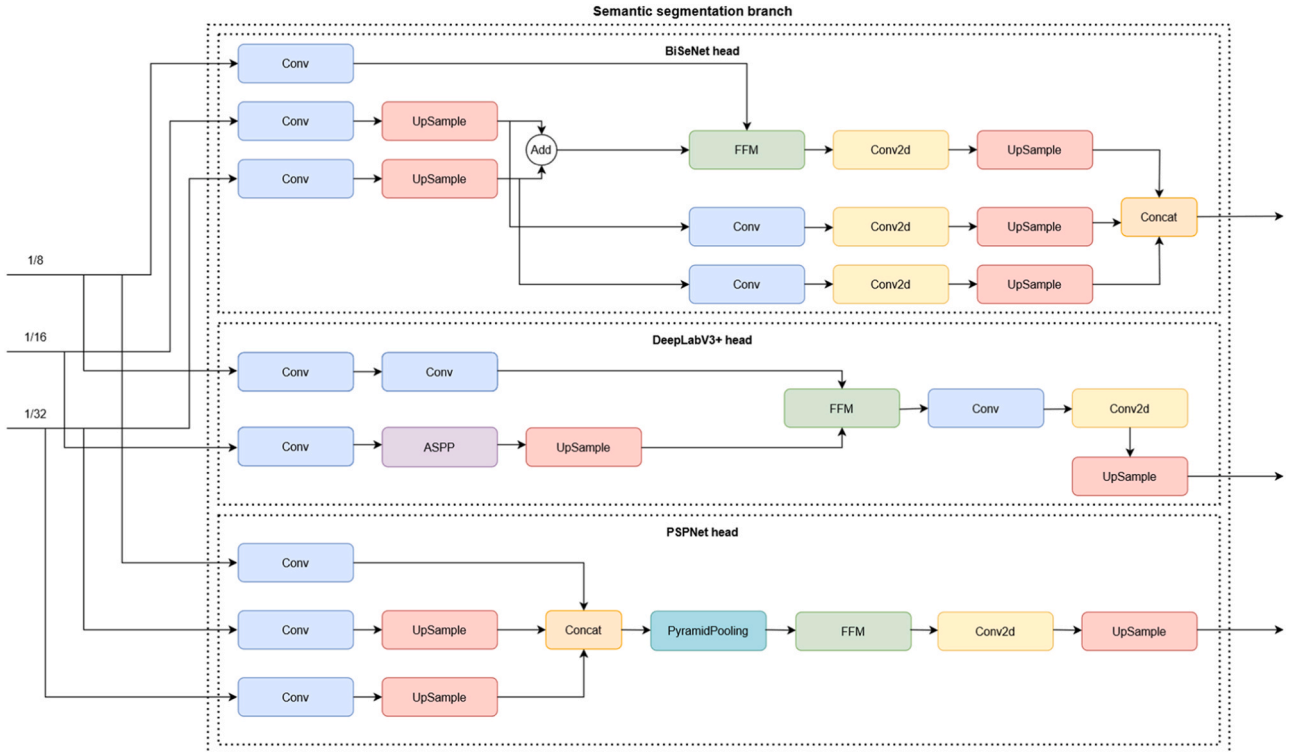


Fig. 6. Overview of the architecture of the 3 types of segmentation heads adapted to our model.

layer and up-sampling by a factor of 2. The ASPP module is a common approach for enlarging receptive fields in the images, hence its usefulness in the DeepLabV3+ head.

PSPNet: the inputs for this head are $1/8$, $1/16$, and $1/32$, lower part in Fig. 6. Prior to Pyramid Pooling (PP), we added a deeper layer to each input. Specifically, we apply a 1×1 convolution to the $1/8$ input, and for the $1/16$ and $1/32$ inputs, we apply a 1×1 convolution followed by up-sampling by a factor of 4 and 2, respectively. Next, we concatenate the results of the three inputs and then apply FFM followed by a 1×1 convolutional layer and up-sampling by a factor of 8. The PP module is also a common approach for enlarging receptive fields in the images, hence its usefulness in the PSPNet head.

3.5. Training: joint update of weights

Once we have defined the network architecture model the weights at all layers must be adjusted. In this respect, mention should be made of the updating strategy.

As usual, the training dataset is split into batches. The images of each batch flow forward through the backbone, neck, and heads (object detection and semantic segmentation) until the output is reached. Then the backpropagation process propagates the loss backward to fine-tune the weights along the head, neck, and backbone.

With each batch, first all weights in the full model are updated through the object detection branch, and then the weights due to the semantic segmentation branch. Therefore, the backbone and neck weights are updated by both processes (object detection and semantic segmentation) while the head weights are only updated by the image segmentation process.

Finally, it should be noted that three types of training are carried out, each corresponding to the three heads of semantic segmentation networks. The updating of the weights in all cases is done from scratch.

4. Experimental results and discussion

Section 4.1 describes the related dataset, including details on data collection, data annotation, and combination of datasets. The experimental configuration is described in Section 4.2. The evaluation metrics used for assessing the performance of the simultaneous model are discussed in Section 4.3. Section 4.4 presents the experimental results and discussion obtained from our image dataset of lentic water scenes. Section 4.5 compares the performance between the reference methods [15] and [17] and our approach on our lentic water dataset.

4.1. Dataset

4.1.1. Related datasets

The objective of the proposed architecture is, as mentioned before, the use of a MSV for simultaneous object detection and semantic segmentation in adverse environments for safe navigation of USVs and of course for detecting and locating amorphous textures, of water bodies, and specifically cyanobacterial patches for monitoring. The intrinsic problems in these outdoor environments have been previously highlighted, so it is necessary to have sufficient datasets to verify the performance of the proposed architecture.

In this regard, we conducted an exhaustive exploration of the various publicly available datasets that include images of aquatic environments. These images show bodies of water and may or may not include objects, captured from the perspective of an ASV. A detailed description of these datasets is presented below:

1. **MaSTr1325** [55]: is characterized by containing 1325 high-resolution images of coastal areas, captured in realistic conditions and some images do not possess any maritime objects.
2. **MODD2** [56]: is also characterized by containing images of coastal areas (28 video sequences of varying length), with the difference that

it possesses complex scenes, such as water reflections, reflections of maritime objects, and fog.

3. **Marine Image Dataset (MID)** [57] and **Singapore Maritime Dataset (SMD)** [58]: are characterized by containing scenes of offshore areas, and a variety of maritime objects, such as buoys, floating objects, and ships.
4. **USVInland** [59]: contains images of inland waterways, it is characterized by containing urban areas (buildings, bridges, roads, etc.) and vegetation. The scenes do not contain objects as such.
5. **INTCATCH** [28]: contains 22 video sequences captured in lakes and rivers. Scenes do not contain objects as such.
6. **RS-CyanoHABs** [33]: contains 2400 images resulting from a combination of two datasets: R-CyanoHABs and S-CyanoHABs. R-CyanoHABs contains real cyanobacterial patches as foreground and images of lakes and reservoirs as background, 1200 images are selected from this dataset. Meanwhile, S-CyanoHABs contains synthetic cyanobacterial patches generated with the Style-based Generative Adversarial Network Adaptive Discriminator Augmentation (StyleGAN2-ADA) [60] approach and refined with Neural Style Transfer [61] as foreground, also using images of lakes and reservoirs as background.

While the datasets described above are diverse in terms of reflections, illumination, and some types of objects, they present the following drawbacks for our purposes, with the exception of the RS-CyanoHABs dataset:

- The scenes of MaSTr1325, MODD2, MID, and SMD contain tides, which means that the water mass is in very noticeable periodic motion that causes sea level rise and fall, which is very different from lentic water dynamics. Meanwhile, the USVInland and INTCATCH datasets do not contain any objects.
- The background scenes of the marine datasets contain sparse vegetation and trees, which is the opposite of that of lentic waters.

Therefore, existing datasets, with the exception of RS-CyanoHABs (which come closest to our purposes), are not suitable for this study, as they lack images that accurately represent lentic water scenes. Consequently, it is imperative to create a specific dataset that captures the unique characteristics of lentic water environments, including images with typical objects of these outdoor environments, and textures such as water bodies, cyanobacterial blooms, among others. Although the RS-CyanoHABs dataset already incorporates cyanobacterial blooms, which allows us to achieve one of the objectives of this study: to facilitate pollutant monitoring, it is still necessary to create an image dataset containing typical objects existing in lentic scenes (canoes, people on board or paddle for navigation, submerged rocks, floating trunks, among others). In this way, with the combination of objects and textures in the dataset, we will be able to achieve the two objectives of this work, i.e., texture identification and object detection for safe navigation of ASVs.

4.1.2. Data collection

Under the above considerations, it is mandatory to generate our own image dataset to effectively meet the objective of containing specific objects and textures of lentic waters. This includes objects such as canoes, persons on board, paddles used for navigation, submerged and emerged rocks, floating tree trunks, among other relevant elements and textures such as water bodies, grasses, trees, mountains, sky, and, of course, patches of cyanobacteria in the water. Of course, considering that all of them must include the effects of the adverse environmental conditions, mentioned above. With this purpose in mind, we have taken the initiative to create our own dataset to effectively meet this objective.

We use images from Flickr [62], a large repository of images whose purpose is to allow people to share their images with other interested parties. These images are stored with detailed information, such as the device used to capture them, camera settings, and location, but they are

unlabeled at the pixel or entity level. To select appropriate images, we apply specific filters using the Flickr API with the arguments and parameters shown in Table 3. In general, each filtered and downloaded image meets the following requirements:

- The image is in the public domain (open access), which means that it is not subject to copyright and can be used for any purpose.
- The image is captured from the canoe or any type of ASV, so they are acquired under perspective projection.
- The image contains at least water and a canoe. Ideally, the image contains water, canoe(s), person(s), paddle(s), submerged rock(s), and floating trunk(s).

The images collected belong to different aquatic environments, most of them are captured in lakes, lagoons, reservoirs, and wetlands, as shown in Fig. 7. There is also a minority of images of rivers and inland waterways. In addition, it is important to note that some images are of open water, where the objects of interest are in the middle of the lake. The background of the images contains a lot of vegetation and trees, which is a common feature of our aquatic outdoor environments, with the associated complexity, mentioned above. Fig. 8 displays some representative examples of images with such complexity: the first row shows scenes with water reflections, reflections of objects of interest, and reflections of surrounding vegetation and trees; the second row shows scenes with different illumination levels.

The size of the collected images is varied, 777 images have a jumbo size ($> 1024 \times 1024$), 108 large sizes indicating medium width by medium height image (1200×800) and 29 medium size images ($< 600 \times 400$), as shown in Fig. 9 and Fig. 10. As for the aspect ratio distribution, 11 images are very tall, 1092 images are square, 518 images are very wide, as shown in Fig. 10 and Fig. 11.

In summary, a dataset consisting of 1622 images was collected, which addresses the lack of data to achieve the main objective, i.e., to detect objects to ensure safe navigation of ASVs and to detect textures for a complete understanding of the scene. Therefore, the collected images contain varied textures, with the exception of cyanobacteria patches, and objects of interest and exhibit a wide range of features, such as variations in quality, size, watercolor, illumination, reflections, weather conditions, and location of objects of interest. As a result, the dataset represents real-world scenes in a sense, capturing the complexities inherent in the task at hand.

4.1.3. Data annotation

To train and evaluate the performance of our proposed architecture, which simultaneously addresses the object detection and semantic segmentation tasks, it is essential to have separate datasets for each of these tasks. In other words, we require one dataset with entity-level annotations specifically for object detection, as well as another dataset with pixel-level annotations targeted at the semantic segmentation task.

In this context, we already have the RS-CyanoHABs dataset, which

plays a crucial role in our study by helping us to fulfill one of the objectives: to detect patches of cyanobacteria in water bodies. The images contained in this dataset are already annotated at the pixel level, which is highly convenient for our purposes. However, we are still missing the annotation of the dataset obtained from the Flickr repository (described in the previous section), which we compiled to complement our research.

For object detection, we annotated the objects of interest in each image using bounding boxes. We established seven classes: canoe, person, paddle, bird, dog, submerged rock, and floating tree trunk. As for semantic segmentation, we annotated the pixel-level amorphous texture areas for the eight classes: water, cyanobacterial blooms, grass, tree, house, mountain, dirt, and sky.

Since image annotation for both object detection and semantic segmentation is a laborious and time-consuming process, we resorted to two state-of-the-art and cutting-edge methods: Grounding DINO [63] and Segment Anything Model (SAM) [64]. These two high-powered Artificial Intelligence (AI) based models significantly speed up and improve the annotation process. Grounding DINO is a zero-shot-based object detector that fuses the concepts of DINO [65] (a transformer-based detector) and Grounded Language-Image Pre-training (GLIP) [66], a pre-training model based on language and vision processing. In other words, Grounding DINO is able to detect arbitrary objects from text that specifies which objects are to be detected in the image. On the other hand, SAM is an image segmentation model that can segment any object present in an image. In this regard, we use the Grounding DINO model to generate bounding boxes around the objects present in the image, and then, SAM takes these bounding boxes as input and converts them into segmentation masks for each object in the image. We then manually validate and refine the resulting annotations.

According to Fig. 12, the number of objects annotated by class is summarized as follows: Trunk (776), Rock (853), Bird (48), Dog (82), Paddle (2375), Person (3366) and Canoe (1855). Therefore, the object detection dataset presents the well-known class imbalance problem, where the classes Bird, Dog, and Trunk are disadvantaged against other classes. To deal with this class imbalance problem we use focal loss [6], just as in YOLOv4 and YOLOv5, which reduces the impact of majority classes on the loss function of the model. After several trial-and-error tests, to fix the tunable focusing parameter, γ , and the weighting α -balanced factor, involved in the focal loss function, we have decided to leave them with the values reported in [6] as the best ($\gamma = 2$ and $\alpha = 0.25$); so that relatively greater importance is assigned to the misclassified examples. In addition, for training, we use the class label smoothing proposed in [67], which converts the hard label to a soft label to obtain a much more robust model according to the different experiments carried out in [20]. This smoothing operation involves two new hyperparameters, which are ϵ and K , so that also after the corresponding trial and error tests, they have been set to the values 0.1 and number of classes (7 for object detection and 9 for semantic segmentation) respectively. In the latter case with a lower value than the one reported in the original reference, since there K is 1000, which is the number of classes used. With different values of all these hyperparameters also fail to achieve noteworthy improvements.

Similarly, the semantic segmentation dataset presents the problem of class imbalance, as shown in Fig. 13. The classes House, Dirt, and Mountain are at a disadvantage compared to other classes. To deal with this problem we also use the focal loss function, with the same hyperparameters and values reported above, without achieving significant improvements with other combinations of values.

After annotating the data with Grounding DINO and SAM, and manually refining it, we proceed to merge real cyanobacterial patches into 500 images. This fusion process is performed using the corresponding masks, ensuring that the patches are accurately integrated on the water surface.

Table 3

Flickr API arguments used for filtering images. Sizes ['url_o', 'url_l', 'url_c'] indicate ['width=2400 height=1800', 'width=1024 height=768', 'width=800 height=600']; privacy filter ['1'] indicates ['public photos'] and content type ['1'] indicates ['for photos only'].

Argument	Description	Value
Text	Returns images whose title, description, or tags contain the text.	'Canoe'
Sizes	Filter results by size	'url_o', 'url_l', 'url_c'
Privacy filter	Returns photos that only match a certain privacy level.	'1'
Content type	Content Type setting.	'1'
Media	Filter results by media type.	'photos'



Fig. 7. Some sample images of lentic waters collected from Flickr: lakes, lagoons, reservoirs, wetlands, rivers, and waterways.

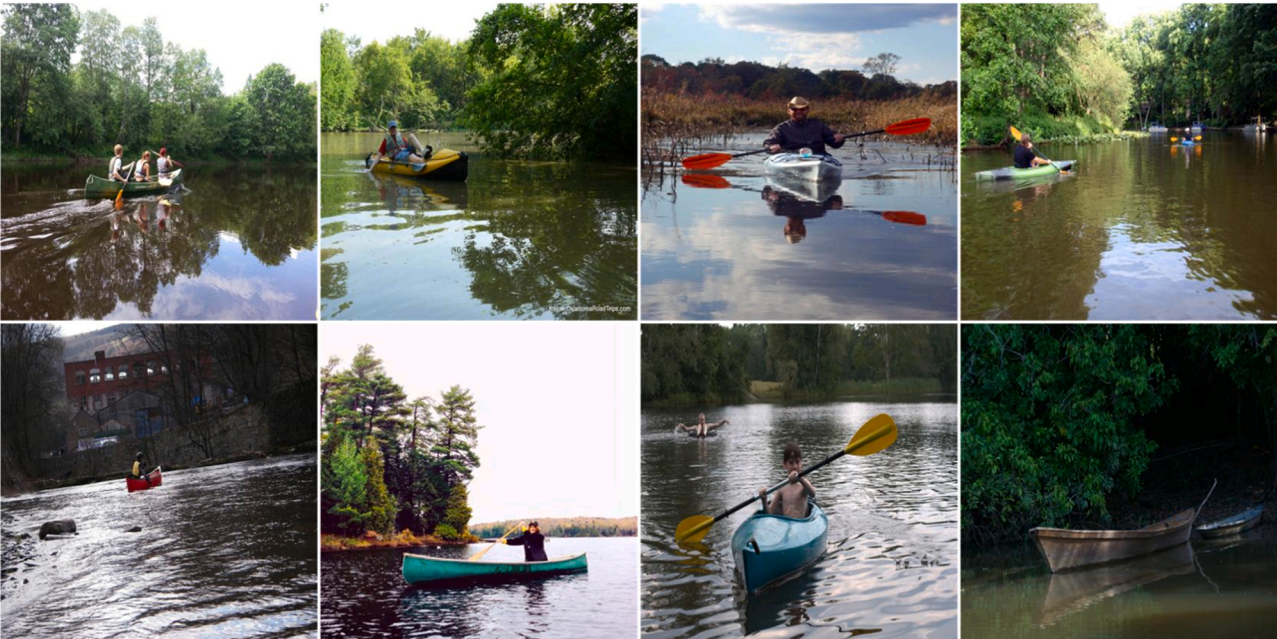


Fig. 8. Some example images of lentic waters. First row: water and object reflections. Second row: different levels of illumination.

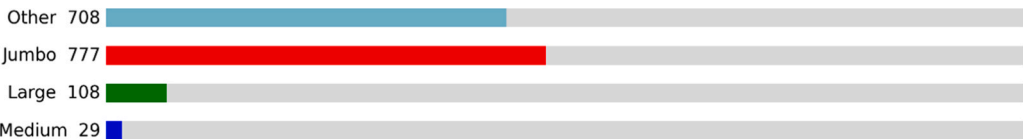


Fig. 9. Image size distribution of the lentic water dataset.

4.1.4. Combination of datasets

The RS-CyanoHABs dataset includes specific annotations for the semantic segmentation task, where pixels have been categorized into four different classes: “blooms” (1), “water” (2), “landscape” (3), and “sky”

(4). On the other hand, images collected from Flickr have been annotated for the two types of tasks: object detection (including categories such as “canoe” (0), “person” (1), “paddle” (2), “dog” (3), “bird” (4), “rock” (5) and “trunk” (6)) and semantic segmentation (with classes

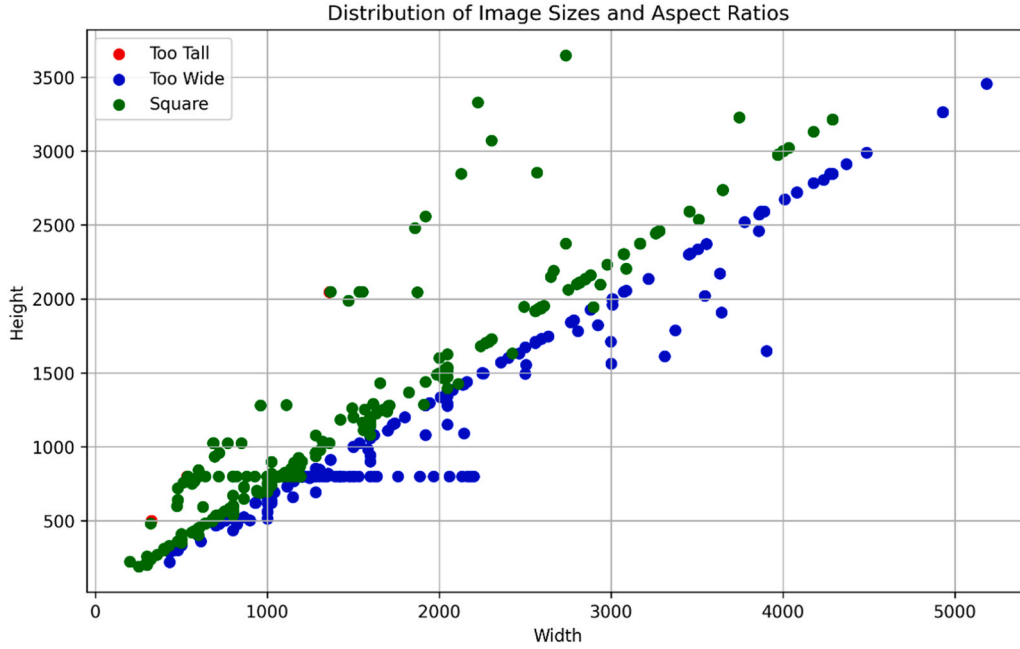


Fig. 10. Size distribution and aspect ratio of images from the lentic water dataset.

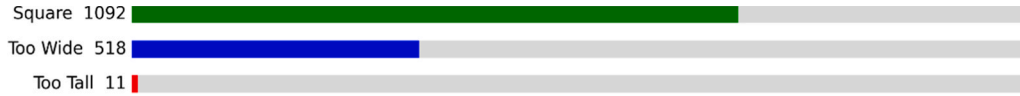


Fig. 11. Aspect ratio distribution of images from the lentic water dataset.

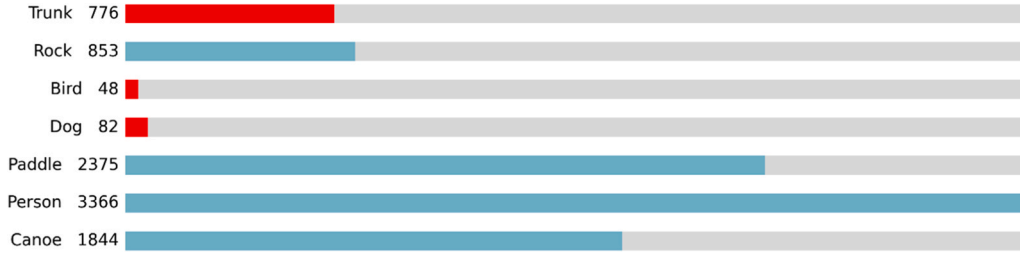


Fig. 12. Class balance of the object detection dataset.

such as “water” (1), “blooms” (2), “grass” (3), “tree” (4), “house” (5), “mountain” (6), “dirt” (7) and “sky” (8)), resulting in two datasets.

It is important to note that these datasets present non-standardized class labels, i.e., the annotations do not follow a common format, as described in the previous paragraph. In this context, we performed the process of standardizing the class labels for both tasks, as can be seen in Fig. 14. The standardization process consists of identifying which labels refer to the same classes and assigning a uniform label to those classes. Then, we merge the standardized datasets with the purpose of creating a final dataset for each task, as shown in the last column of Fig. 14.

In detail, we approached the construction of the final dataset for the object detection task by combining two datasets (RS-CyanoHABs and images collected from Flickr) previously intended for this task. One challenge we faced was that one of the sets, RS-CyanoHABs, lacked entity-level annotations due to the absence of objects of interest. In response, we decided to incorporate null annotations, a choice informed by the nature of lentic water scenes, where objects are not always present. This inclusion of null annotations has proven beneficial by reducing the false positive (FP) rate, which, in turn, improves the accuracy of the dual-task model.

Regarding the construction of the final dataset for the semantic segmentation task, we also chose to combine two datasets (RS-CyanoHABs and images collected from Flickr) specifically designed for this purpose. Previously, we standardized the class labels to ensure that the annotations followed a uniform format, as detailed in Fig. 14.

Merging these datasets has allowed us to create a coherent and complete final dataset, as illustrated in Fig. 14, which is suitable for both object detection and semantic segmentation. In addition, we performed a random partitioning into hyperparameter setting (12 %), training (80 %), and validation (20 %) sets, as shown in Table 4, indicating the number of images in each case.

4.2. Experimental configuration

All experiments were conducted on the Google Colab platform which provides an NVIDIA Tesla T4 GPU with 16 GB of memory. The proposed dual-task model has been implemented using the PyTorch 2.0.1 Deep

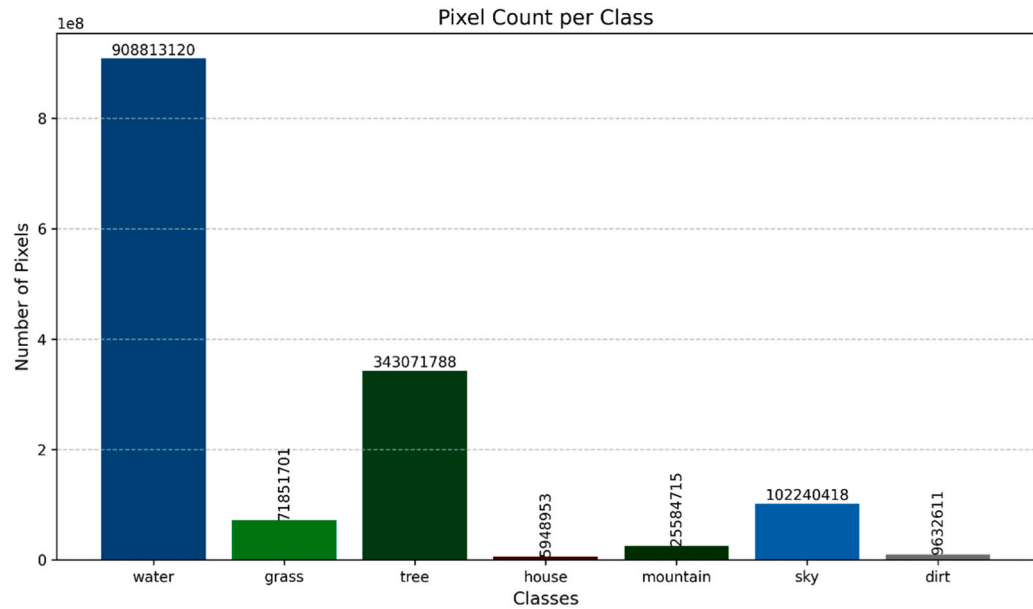


Fig. 13. Class balance of the semantic segmentation dataset.

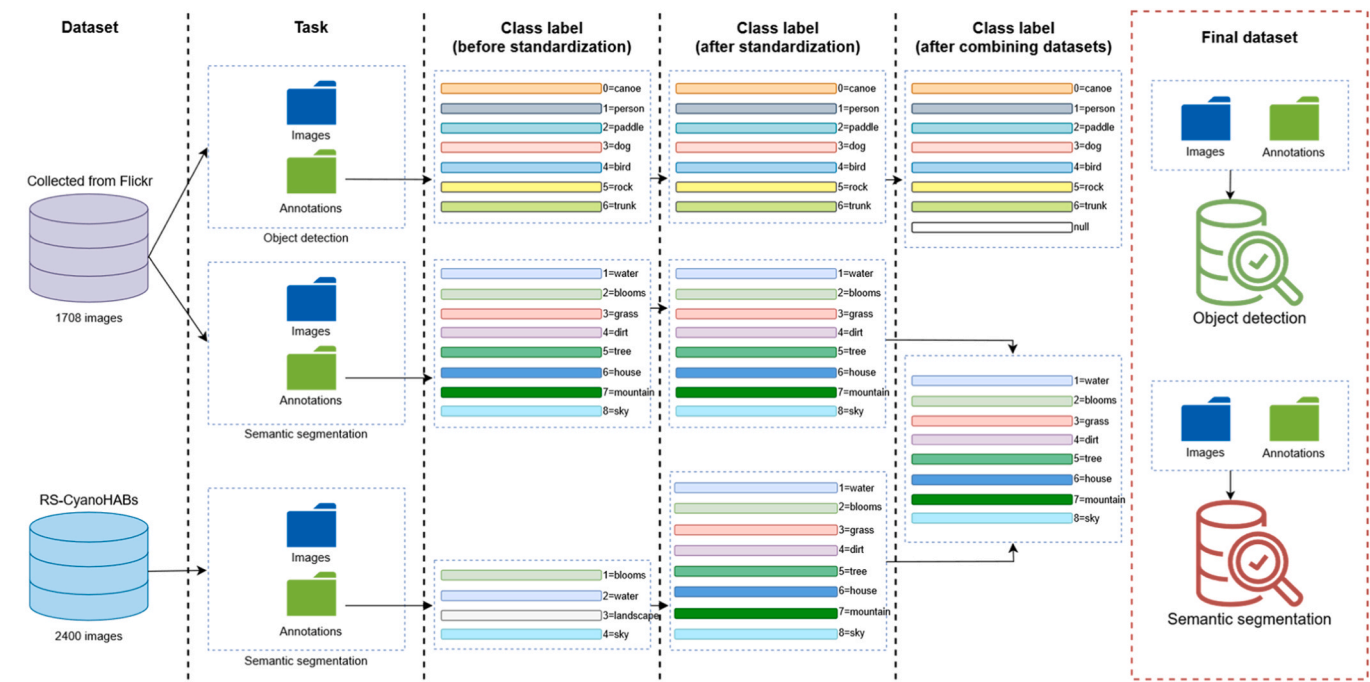


Fig. 14. The process of standardization of class labels.

Table 4
Split of the datasets.

Dataset	Hyperparameter setting (12 %)	Training (70 %)	Validation (18 %)
Object detection	398	3286	822
Semantic segmentation	398	3286	822

learning framework and Python 3.10.12 as the programming language. It is based on YOLOv5 tag v5.0 implemented natively in PyTorch. The source code is available on GitHub,¹ containing two core modules where both the object detector and the semantic segmentation are implemented.

4.3. Evaluation metrics

To evaluate the performance of object detection models, the mean average precision (mAP) is used, which takes values in the range [0,1].

¹ <https://github.com/fbarrientos/multitask/>

This metric measures the ability of the model to correctly predict bounding boxes with a certain confidence level (usually mAP@0.5 or mAP@0.95). The confidence level indicates the percentage of area that a bounding box overlaps with the ground truth boundary. To evaluate the performance of semantic segmentation models, the mean intersection over union (mIoU) is used. Therefore, in this study, we use mAP with a confidence level of 0.5 as the metric to evaluate the object detection task and mIoU as the metric for semantic segmentation.

With the data selected for hyperparameter setting as the training dataset and using the validation dataset, Table 4, we have trained in a joint manner end-to-end using the Stochastic Gradient Descent (SGD) optimizer with momentum [68] against Adaptive Moment Estimation (Adam) [69]. To determine the initial learning rate at this stage we use a learning range finder test by applying a grid search strategy based on a generalization criterion. The generalization is measured as the lowest error differences between training and validation dataset errors, as explained below. We create a grid, in a logarithmic scale, of learning rates [5e-1, 5e-2, 5e-3, 5e-4] and decays [5e-2, 5e-3, 5e-4, 5e-5] to explore the best generalization of SGD and Adam. Also, three momentum's values [0.89, 0.90, 0.91] are set for SGD and Adam (first momentum), considering that 0.90 is a typical value used in different works, including [69]. In Adam, we have set the second moment to 0.99, which is also a typical value used in different experiments. The maximum number of epochs has been set to 120. Considering the 48 combinations of the values of these hyperparameters, both training processes (object detection and semantic segmentation, Section 3.5) are launched to reach the maximum number of epochs. The best combination of hyperparameters is determined by computing relative minimal differences between the two training and validation losses once the underfitting zone is passed and before reaching the overfitting zone, identifying the number of epochs where this point of interest occurs. The differences are averaged values between the two loss functions in the head, i.e. in both object detection and semantic segmentation branches. It is worth mentioning that this critical point appears around 85 epochs with a batch size of 16 (i.e. 16 images at each iteration with 25 iterations per epoch considering the 398 training images), which are the values finally chosen. Different experiments have been carried out by varying the batch size (8, 16, 32, 64) without obtaining any relevant improvement in the performance of the results. We have verified that SGD with a learning rate of 5e-2, decay 5e-4, and momentum of 0.90 generalizes better than Adam, although Adam converges faster. Considering generalization as the most relevant criterion, we chose SGD as the final optimizer for the experiments with 5e-2 as the benchmark learning ratio.

We apply the symmetric OneCycleLR [70], policy for fine-tuning the hyperparameters in a cyclical manner with two steps of similar lengths, the first going from a lower learning rate to a higher one and the second going back to the minimum (decay phase), allowing the learning rate to decrease beyond the minimum by one order of magnitude. The maximum is the benchmark learning rate (i.e. 5e-2) and the minimum is ten times lower (i.e. 5e-3). The length of the cycle is slightly less than the number of epochs (80) with 2000 iterations. The momentum varies from 0.80 to 0.90 in the first half of the cycle and then remains fixed during the second half. As expected, and according to [70], we have verified that in the middle of the cycle, the model is kept from overfitting achieving the best performance with a learning rate of 1.4e-3 beyond the end of the cycle around 100 epochs. This is the point where the differences between the training loss and the validation loss achieve the minimum differences after the oscillations of the previous iterations and before the divergence occurs due to the entry into the overfitting zone. Thus, considering the decay factor established above ($d = 5e-4$), with a number of epochs ($e = 100$) and the learning rate schedule as $\alpha = \alpha_0 / (1 + d \times e)$, the initial learning rate α_0 is 1.5e-3.

As a general conclusion derived from the above strategies (learning range finder and OneCycleLR) is that the mode of choice of the hyperparameters is robust enough and efficient. Indeed, the evolution of the learning rate from its initial value (1.5e-3) to its best value (1.4e-3) is

very slow, confirming the relatively low influence of the learning rate schedule and the good selection of this hyperparameter by the OneCycleLR process. Moreover, they are values within the ranges and orders of magnitude used in many convolutional neural network models, so their use is valid in the scenarios that concern us and preserve the trend of different experiments reported in the literature.

In addition, the model leverages pre-trained weights from the yolov5s model, the second smallest and fastest model, which was previously trained on the MS COCO dataset [71].

Moreover, we apply the strategy of the different experiments proposed in [20] to improve the overall performance of the model, namely: the mosaic data augmentation, which mixes four input images to teach the model to find the smallest objects of interest; the class label smoothing; the Mish [72] activation feature, which avoids the gradient fading problem; the DropBlock [73] regularization method, which requires the model to keep learning features when a part of the object is hidden; and Self-Adversarial Training (SAT), which operates in two stages forward and backward; the CIoU [53] loss function, which sees how the predicted bounding box overlaps with the ground truth bounding box.

4.4. Experimental analysis

In this section, firstly we present quantitative experimental performance results of the model for both, object detection and semantic segmentation (dual-task model) on validation datasets containing images with lentic water scenes. Specifically, we evaluate the impact of the CBAM attention mechanism on both the overall performance and the per-class performance of the model. As mentioned before, CBAM is a relevant structure, focusing on where and how the information is relevant and is also the gateway for information from the neck to the detector heads. This is the reason for checking its relevance and performance with the intended analysis of this attention mechanism. In this regard, we designed two sets of experiments on the dual-task model with and without CBAM. Secondly, quantitative experimental performance results are also discussed considering single-task models for both, object detection and semantic segmentation. Finally, we perform a qualitative analysis of the performance of the dual-task model versus single-dual task models and extend this analysis to consider some of the challenges that our approach successfully addresses.

4.4.1. Results of the dual-task model without CBAM

Different experiments and configurations have been applied without CBAM. In this regard, the CBAM modules in Fig. 4, have been disconnected from the neck and the outputs towards the heads are directly connected from the corresponding three C3 modules. For each experiment, we use the same backbone (CSPDarknet-53) and the same object detection method (YOLOv3 head used in YOLOv4 and YOLOv5) in the head, and for the semantic segmentation branch, the three segmentation heads (BiSeNet, DeepLabV3+, and PSPNet) have been tested. Here we are interested in finding out whether the proposed dual-task mechanism benefits the object detection and semantic segmentation task, especially which of the segmentation head types benefits the object detection branch the most and vice versa.

Table 5 displays the quantitative performance results of the dual-task model without CBAM on the image validation dataset with lentic water scenes, indicating: a) the classes of interest for both object detection and semantic segmentation, and b) columns 2–6 the performance results.

In the following, we remark on some points derived from Table 5, specifically for the object detection branch:

- The results of the object detection branch, when effectively combined with the results of the semantic segmentation branch (BiSeNet, DeepLabV3+, and PSPNet) show good overall performance, the difference between the best and worst mAP score is only 3.0 %

Table 5

Results of the dual-task model of object detection and semantic segmentation without CBAM. The heads for object detection is YOLOv3, used in YOLOv4 and YOLOv5 and the heads for semantic segmentation are BiSeNet, DeepLabV3+, and PSPNet.

Classes	YOLOv3 and BiSeNet		YOLOv3 and DeepLabV3+		YOLOv3 and PSPNet	
	mAP@.5	mIoU	mAP@.5	mIoU	mAP@.5	mIoU
Object detection branch						
Canoe	85.41	—	87.31	—	86.71	—
Person	94.90	—	96.00	—	96.60	—
Paddle	67.21	—	69.11	—	69.90	—
Dog	33.41	—	34.60	—	32.91	—
Bird	23.79	—	22.20	—	34.50	—
Rock	28.59	—	30.81	—	33.01	—
Trunk	28.30	—	29.90	—	29.00	—
All	51.66	—	52.85	—	54.66	—
Semantic segmentation branch						
Water	—	92.31	—	93.41	—	93.61
Blooms	—	88.22	—	88.42	—	89.32
Grass	—	50.33	—	51.55	—	54.23
Dirt	—	39.90	—	45.00	—	39.50
Tree	—	85.00	—	85.69	—	85.80
House	—	44.61	—	45.51	—	46.70
Mountain	—	38.81	—	37.12	—	43.72
Sky	—	92.20	—	94.30	—	93.81
All	—	66.42	—	67.63	—	68.34

(54.66 %-51.66 %). This indicates that the object detection branch generalizes very well on the validation dataset.

- The object detection branch achieves its maximum overall performance benefiting from the proposed combined mechanism, with the PSPNet segmentation head standing out, followed by DeepLabV3+ and BiSeNet, i.e., PSPNet provides the best global and local semantic information to the object detection branch than the other segmentation heads. For example, segmentation tells object detection that canoes and paddles should be on the surface of the water and not on the landscape or in the sky.
- The object detection branch, when combined with the semantic segmentation branch based on the PSPNet head obtains the best results for the following classes: person, paddle, bird, and rock. While for the classes canoe, dog, and trunk, the combination with the DeepLabV3+ head obtains the best results.

According to studies [14–16], the fact that the head-end branches are fed by the same feature maps, coming from the neck, allows a simultaneous adjustment of the network weights leading to a mutual benefit derived from the shared information, obtaining better performance than the individual detection or segmentation networks.

On the other hand, regarding the semantic segmentation branch, the analysis of results from Table 5 can be summarized as follows:

- The three segmentation branches (BiSeNet, DeepLabV3+, and PSPNet) with YOLOv3 obtain good overall performance, the difference between the best and worst mIoU score is 1.92 % (68.34 %-6.426 %). Because as the weights are first adjusted by the object detector, some pixels classified by the semantic segmentation are affected by this fact. Thus, for example, the pixels below the canoe tend to be classified as areas of the water's surface.
- The semantic segmentation branch based on the PSPNet head obtains the best overall score, followed by DeepLabV3+ and BiSeNet.
- PSPNet obtains the best results for the six classes: water, blooms, grass, tree, house, and mountain. Meanwhile, DeepLabV3+ outperforms the two others for the dirt and sky classes. This suggests that, in general terms, PSPNet generalizes well for textures that are more amorphous and with a higher granularity, although the other two models also achieve acceptable performances, and the three models differentiate well from the textures analyzed.

4.4.2. Results of the dual-task model with CBAM

The CBAM structures are preserved as shown in the architecture shown in Fig. 4. Therefore, the outputs of the three CBAM modules feed the heads of the object detector and the semantic segmentation module. As before and as the architecture is designed, for each experiment we use the same backbone (CSPDarknet-53), the YOLOv3 head for object detection, and BiSeNet, DeepLabV3+, and PSPNet for semantic segmentation.

Table 6 shows the results of the dual-task model of object detection and semantic segmentation after adopting the CBAM mechanism.

In terms of mAP by class, we can infer an improvement in results, with respect to the structure without CBAM. Considering the best results for each case, in the object detection branch with YOLOv3 head. The class “trunk” experienced a remarkable improvement of 3.92 %, followed by the class “paddle” with an increase of 3.28 %, the class “rock” with an increase of 2.45 %, the class “canoe” with an increase of 2.30 %, the class “person” with an increase of 1.01 %, the class “dog” with an increase of 0.65 %, and the class “bird” with an increase of 0.41 %. On the other hand, in the semantic segmentation branch, the “blooms” class showed a remarkable improvement of 1.44 %, followed by the “house” class with an increase of 1.14 %, the “dirt” class with an increase of 0.92 %, the “water” class with an increase of 0.72 % and “grass” class with an increase of 0.68 %, the “tree” and “sky” classes with an increase of 0.41 %, and the “mountain” class with an increase of 0.09 %. Therefore, we can clearly see that CBAM improves the accuracy of most classes and, consequently, the overall performance of the model.

In summary, the performances of the object detection branch and semantic segmentation are slightly improved on our image dataset of lentic water scenes, outperforming the dual-task network without CBAM by 1.37 % in detection and 0.66 % in semantic segmentation. These experimental results (see Table 6) show that our dual-task model with CBAM is much more effective in the object detection branch, and this indicates that CBAM induces the model to correctly focus on objects of interest (navigational obstacles) that are in adverse environmental conditions. Furthermore, from Table 6, it can be inferred that CBAM also contributes to the detection of small objects found in regions affected by adverse environmental conditions, such as “trunk”, “paddle”, “rock”, etc. In contrast, regarding the semantic segmentation branch, the improvement is less remarkable (0.66 %). This is because the amorphous semantic textures addressed in this study generally represent

Table 6

Results of the dual-task model object detection and semantic segmentation with CBAM. The heads for object detection is YOLOv3, used in YOLOv4 and YOLOv5, and the heads for semantic segmentation are BiSeNet, DeepLabV3+, and PSPNet.

Classes	YOLOv3 and BiSeNet		YOLOv3 and DeepLabV3+		YOLOv3 and PSPNet	
	mAP@.5	mIoU	mAP@.5	mIoU	mAP@.5	mIoU
Object detection branch						
Canoe	87.32	—	89.61	—	88.09	—
Person	96.84	—	96.70	—	97.61	—
Paddle	67.31	—	71.27	—	73.18	—
Dog	34.10	—	35.25	—	33.74	—
Bird	26.15	—	23.32	—	34.91	—
Rock	31.00	—	32.90	—	35.46	—
Trunk	32.22	—	30.41	—	29.21	—
All	53.56	—	54.21	—	56.03	—
Semantic segmentation branch						
Water	—	92.92	—	94.02	—	94.33
Blooms	—	89.37	—	89.71	—	90.76
Grass	—	50.60	—	52.48	—	54.91
Dirt	—	40.91	—	45.92	—	40.50
Tree	—	85.05	—	86.10	—	85.82
House	—	45.11	—	45.76	—	47.84
Mountain	—	38.83	—	37.64	—	43.81
Sky	—	93.12	—	94.72	—	94.00
All	—	66.99	—	68.29	—	69.00

large regions, such as the “mountain” region, where the improvement barely reaches 0.09 %, or “tree”, with an increase of 0.41 %, etc. Overall, the experimental results demonstrate the effectiveness of the CBAM mechanism in improving the overall model performance.

4.4.3. Results of the dual-task model versus single-task models

After verifying the outperformance of the proposed dual-task model with CBAM versus the model without CBAM, we extend the experiments to analyze the behaviors of the single-task methods in our dataset. The comparison is based on the models with and without CBAM. The following considerations have been taken into account in the choice of the single-task methods, which were trained and evaluated with our object detection and semantic segmentation dataset:

- 1) Object detection: two families have been studied. Firstly, that is based on the one-stage approach where the entire image is processed in a single step. YOLOv5, SSD (Single Shot Multibox Detector), and RetinaNet are the ones selected. YOLOv5 outperforms the other two, being the one selected. On the other hand, methods of the R-CNN family, including Fast, Faster, and Mask, have been studied, as representatives of two-stage strategies. Among them, Faster R-CNN is chosen for its outperformance.
- 2) Semantic segmentation: the study of the same strategies (BiSeNet, DeepLabV3+, and PSPNet) separately leads to the fact that the best performance is obtained with PSPNet, in line with its behavior in the Dual task model, according to the results reported in [Tables 5 and 6](#). This is the reason for reporting the PSPNet results as single.

[Table 7](#) summarizes the different results obtained with the single (R-CNN and YOLOv5) and dual-task (YOLOv3-PSPnet with and without CBAM) models, distinguishing between object detection and semantic segmentation.

According to the results in [Table 7](#), we can infer that the results obtained with the Dual-task model without CBAM improve the results of the single tasks and, by extension, those obtained with CBAM even more ([Table 6](#)). Going into a bit more detail on the object detection task, our model with CBAM shows significantly superior performance compared to the model without CBAM, as well as to the single-task models YOLOv5, and Faster R-CNN, obtaining a mAP of 56.03 % versus 54.66 %, 52.22 % and 48.17 %, respectively. It is worth noting its remarkable performance in the detection of the “person” class, reaching

an accuracy of 97.61 %, outperforming the model without CBAM (96.60 %), YOLOv5 (82.50 %), and Faster R-CNN (80.92 %). As for the semantic segmentation task, our dual-task approach with CBAM achieves an overall performance of 69.00 %, outperforming the model without CBAM (68.34 %) and the single-task model PSPNet (66.26 %). Although PSPNet obtained better results in the “tree” class, our method with CBAM demonstrated superior performance in most classes, especially standing out in “water” and “sky” with accuracies of 94.33 % and 94.00 % respectively. These results demonstrate the effectiveness of our dual-task method and in particular with CBAM, which acts as an enhancer of the joint model for tackling computer vision tasks in lentic aquatic environments, i.e. outperforming models without CBAM and those focused on single tasks.

[Table 8](#) shows a comparative summary between the results obtained by our dual-task model and the single-task models derived from [Tables 5, 6, and 7](#).

Regarding the object detection task, our dual-task model, incorporating the CBAM-based attention mechanism, provides superior performance (56.03 %) compared to other models, such as the dual-task model without CBAM (54.66 %) and the single-task models (52.22 % and

Table 8

Comparison of the dual-task model (focusing on object detection and semantic segmentation) and the single-task models of object detection and semantic segmentation.

Mode	mAP@.5	mIoU
Object detection		
Single-task (YOLOv5)	52.22 (Table 7)	—
Single-task (Faster R-CNN)	48.17 (Table 7)	—
Dual-task (object detection branch) without CBAM	54.66 (Table 5)	—
Dual-task (object detection branch) with CBAM	56.03 (Table 6)	—
Semantic segmentation		
Single-task (PSPNet)	—	66.26 (Table 7)
Dual-task (semantic segmentation branch) without CBAM	—	68.34 (Table 5)
Dual-task (semantic segmentation branch) with CBAM	—	69.00 (Table 6)

Table 7

Comparison of the results of our dual-task method with and without CBAM (YOLOv5 and PSPNet as heads for object detection and semantic segmentation, respectively) with single-task methods (Faster R-CNN, YOLOv5, and PSPNet) on our lentic water imaging dataset.

Classes	Single-task model			Dual-task model		
	Faster R-CNN [41]	YOLOv5	PSPNet [10]	YOLOv3 and PSPNet (with and without CBAM)		
	mAP@.5		mIoU	mAP@.5	mIoU	
Object detection						
Canoe	74.10	80.02	—	88.09	86.71	—
Person	80.92	82.50	—	97.61	96.60	—
Paddle	62.16	68.34	—	73.18	69.90	—
Dog	31.70	31.22	—	33.74	32.91	—
Bird	33.74	32.76	—	34.91	34.50	—
Rock	29.98	30.28	—	35.46	33.01	—
Trunk	24.62	26.40	—	29.21	29.00	—
All	48.17	52.22	—	56.03	54.66	—
Semantic segmentation						
Water	—	—	90.20	—	94.33	93.61
Blooms	—	—	89.34	—	90.76	89.32
Grass	—	—	52.22	—	54.91	54.23
Dirt	—	—	39.87	—	40.50	39.50
Tree	—	—	85.94	—	85.82	85.80
House	—	—	40.62	—	47.84	46.70
Mountain	—	—	41.70	—	43.81	43.72
Sky	—	—	90.21	—	94.00	93.81
All	—	—	66.26	—	69.00	68.34

48.17 %). Regarding the single-task approaches, YOLOv5 outperforms Faster R-CNN. Specifically, a significant improvement in the mAP value is seen, increasing from 48.17 % to 56.03 %. These results confirm that the object detection task benefits from the incorporation of the semantic segmentation task. In other words, semantic segmentation provides contextual information and semantic features. As for the semantic segmentation task, our dual-task model with CBAM (69.00 %) also demonstrates superior performance compared to other models, such as the dual-task model without CBAM (68.34 %) and the PSPNet single-task model which achieves 66.26 % (greater than the ones obtained by BiSeNet and DeepLabV3+). Specifically, we observe a remarkable improvement in the mIoU value, increasing from 66.26 % to 69.00 %. This correlation between the two tasks involved in the dual-task model has been supported by other studies [14–16], which also show that both tasks benefit each other because they share the same feature map extraction network, and forward and backward propagation updates the model parameters causing the model to learn parameters that simultaneously benefit the detection task and the segmentation task.

4.4.4. Qualitative analysis based on visual inspection

Considering the previous quantitative results, we now turn to some qualitative considerations. Firstly, Fig. 15 illustrates some visual results obtained by our dual-task model with and without CBAM and single-task models on four representative images of the dataset, so that what is expressed on them is valid for the total dataset used in our experiments. The first row contains the original images. From the second to the fourth row, the results were obtained with the dual-task model with CBAM, object detection, semantic segmentation, and combined results respectively. The fifth row shows the results of the YOLOv5 single-task model due to its better performance versus Faster R-CNN (Tables 7 and 8). YOLOv5 predicts the objects of interest fairly well in terms of bounding boxes and class categorization. However, in the fifth image of that row, misclassification is observed in predicting the vegetation shadow as the "trunk" class. In contrast, our dual-task model with CBAM (object detection branch) tends to make more accurate predictions in general, indicating that the semantic segmentation branch provides the global and local semantic information to the object detection branch, obtaining superior performance to the single-task object detection models. The sixth row displays the results obtained by PSPNet single-task model. It predicts semantic structures of amorphous nature with relative accuracy. However, in images three and four of that row, misclassification of certain pixels is observed. In contrast, our dual-task model with CBAM (semantic segmentation branch) tends to generate more accurate predictions in general, producing sharp boundaries in segments of amorphous nature. The seventh row shows the results obtained by the dual-task model without CBAM. Visually and qualitatively, there are no significant appreciable differences between the dual-task models with and without CBAM, except for the edges of the textured regions and objects, which appear better outlined when CBAM is used.

Secondly, Fig. 16 shows some illustrative results of the integration of object detection and semantic segmentation, with the aim to show some remarkable aspects of the effectiveness of our dual-task method with CBAM. Four columns (A, B, C, D) are identified with four rows each. The first row corresponds to the input images, the second contains object detection results, the third the semantic segmentation results, and the fourth the integrated results of object detection and semantic segmentation. The following comments are specific for the images shown, being fully valid for the entire set of images analyzed, allowing qualitative inferences about the improvement and effectiveness of the dual approach compared to the individual methods in addressing the specific challenges present in environments such as lentic water scenes. Table 9 summarizes the analysis derived from Fig. 16, addressing only the challenges associated with object detection and semantic segmentation using a dual-task approach. In this regard, objects (people, canoes, etc.) and textures that do not cause difficulties are not considered from the point of view of this challenge.

- Image A, the objects of interest (rock and trunk) are located within a region affected by adverse environmental conditions (low illumination and shadows) caused by the surrounding vegetation. These objects, which are small and distant, are detected with remarkable accuracy (second row). As for the textures, it is observed that a part of the cyanobacterial blooms is in a region characterized by low illumination and shadows. They are segmented with high accuracy under such conditions (third row). Furthermore, we can observe that our approach produces sharp boundaries in semantic segments of amorphous nature.
- Image B, the objects of interest (rock, trunk), are small and far away, where reflections produced by the surrounding water environment are evident. The second row shows how they have been detected. Amorphous textures with sharp boundaries are segmented.
- Image C, the object of interest (trunk far away) is in a region with poor illumination caused by the surrounding vegetation, as well as reflections from the water. Here the canoe object appears with an unusual shape. Our method successfully addresses these challenges by detecting the objects with remarkable accuracy. As for the amorphous textures, such as water, dirt, grass, and tree with low illumination and reflections, our method overcomes these challenges by achieving segmentation with accuracy.
- Image D, the water has a cloudy coloration, which is not typical with strong reflections. These challenges are successfully addressed, and the objects accurately detected. The amorphous textures and the greyish coloring of the sky are correctly identified, also with sharp boundaries.

To conclude the previous analysis, we would like to point out that qualitatively it is difficult to distinguish between results obtained with and without CBAM in the dual-task model. As mentioned above, overall, when CBAM is used, better profiles appear on the edges that delimit textures and objects, which confirms its capacity to capture spatial contextual information at each scale, as expected.

4.5. Extension of comparative performance

As an extension to the tests previously carried out, and to perform a comparative analysis with other object detection and joint semantic segmentation strategies, we have selected the benchmark models proposed in [15] and [17]. All of them have been tested with the dataset described above in Section 4.1. These models are introduced in Section 2.3 and explained below. The comparison in the semantic part is established with PSPNet whose performance is very relevant with respect to DeepLabV3+ and BiSeNet according to the results displayed in Table 6.

Regarding [15] its PriorBox mechanism is designed for detecting objects at several scales through different ratios for each feature map location. In our proposal, this same objective is achieved in a more efficient way by diversifying the information across the different scales in the Neck, and by introducing the specific SPP and PAN modules, including CBAM, which are in charge of capturing the information at the different resolution levels to provide it to the object detector in the Head. On the other hand, the MSAC in [15] captures semantic information, based on atrous convolutions, which is just the foundation of DeepLabV3+. Although DeepLabV3+ performs well in our approach, it is PSPNet that also achieves very good results, Table 6, demonstrating in this case that atrous convolution can be improved.

In [17] the detection branch is based on SSD [4], which uses multiple layers of feature maps with different scales to detect objects at various sizes. In our approach, this is conveniently handled in the Neck, as explained before. The segmentation branch is inspired by PSPNet, which is one of our proposals.

An important aspect to emphasize in our model is the interrelationships (interchannel/interspatial) that appear at different levels of resolutions, which introduce an essential differential nuance

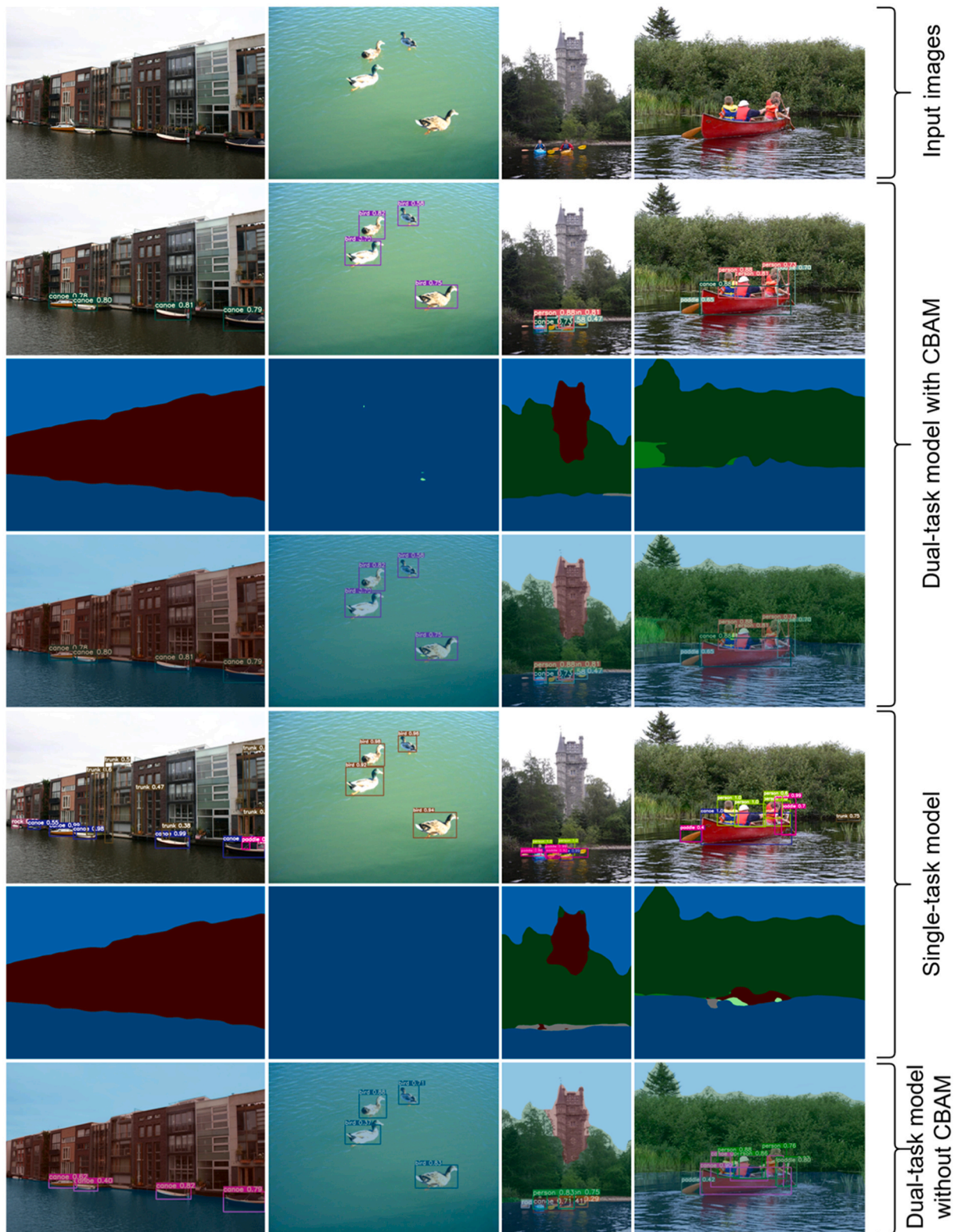


Fig. 15. Visual presentation of the experimental results obtained by our dual-task model with and without and the single-task models. First row: input images. Second row: object detection results using the dual-task model with CBAM. Third row: semantic segmentation results using the dual-task model with CBAM. Fourth row: integrated results of object detection and semantic segmentation of the dual-task model with CBAM. Fifth row: object detection results using YOLOv5. Sixth row: semantic segmentation results using PSPNet. Seventh row: integrated results of object detection and semantic segmentation of the dual-task model without CBAM.

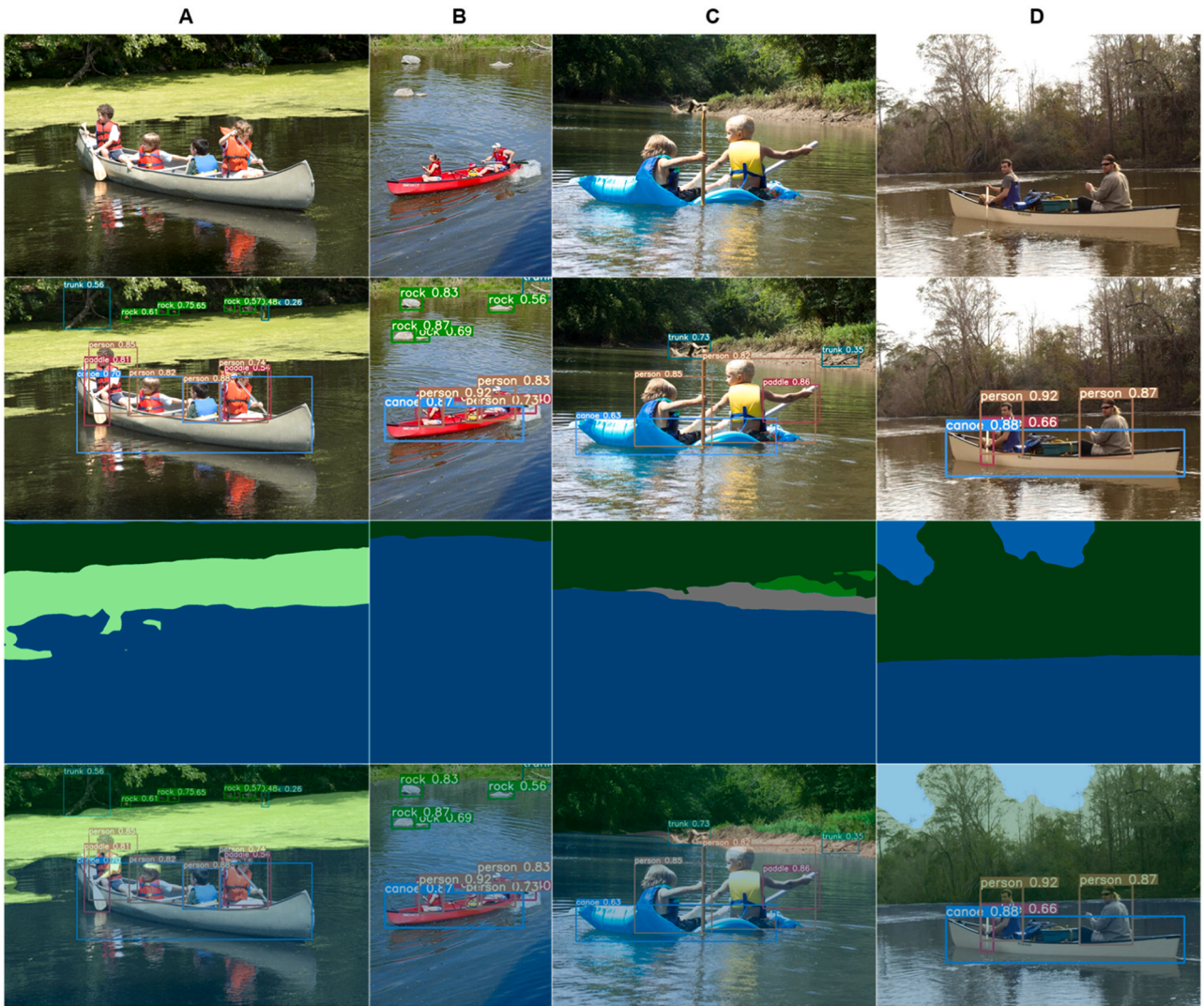


Fig. 16. Visual presentation of the experimental results obtained by our dual-task model with CBAM. First row: input images with challenges. Second row: object detection results. Third row: semantic segmentation results. Fourth row: integrated results of object detection and semantic segmentation.

Table 9

Addressing the challenges associated with object detection and semantic segmentation using the dual-task approach with CBAM: summary of the analysis from Fig. 16.

Challenge		Achievements	
Object detection	Semantic segmentation	Accurate object detection	Accurate semantic segmentation with sharp borders
A	Low illumination, shades, small and distant objects	✓	✓
B	Water reflections, small and distant objects	✓	✓
C	Water reflections, low illumination, distant objects, non-typical canoe	✓	✓
D	Cloudy water, water reflections, gray sky	✓	✓

with respect to [15] and [17]. Both references use outdoor traffic scenes. The fact that in all cases, the datasets are outdoor scenes adds another justification for comparison.

Table 10 shows a comparison of performance between the benchmark methods and our approach on the lentic water dataset. For object detection, our dual-task method stands out for its superior performance on most object classes, including “canoe”, “dog”, “bird”, “rock”, and “trunk”. The main strength of our method lies in the integration of several modules. In particular, the SPP module stands out for its ability to capture relevant information at different scales, thus facilitating the accurate detection of objects of various sizes. For its part, the CBAM module maximizes the exploitation of inter-channel and inter-spatial relationships of features, focusing attention on relevant properties of objects, which is crucial for improving the accuracy of object detection. Finally, the PAN module plays an essential role by integrating the information of all the enhanced feature maps at different scales and resolutions, coming from objects of different sizes.

As for the semantic segmentation results, they are also presented in Table 10. Our method shows outstanding performance in most of the semantic classes, such as “water”, “blooms”, “grass”, “dirt”, and “mountain”. This outstanding performance of our approach can be attributed, in part, to the inclusion of the aforementioned modules. Specifically, the SPP module facilitates the accurate identification and

Table 10

Comparison of the results of our dual-task method (YOLOv3 and PSPNet) with the reference methods on our lentic water imaging dataset.

Classes	Peng et al. [15]		DSPNet [17]		YOLOv3 and PSPNet	
	mAP@.5	mIoU	mAP@.5	mIoU	mAP@.5	mIoU
Object detection branch						
Canoe	87.71	—	80.10	—	88.09	—
Person	97.80	—	92.31	—	97.61	—
Paddle	73.20	—	61.51	—	73.18	—
Dog	32.42	—	28.60	—	33.74	—
Bird	28.30	—	21.39	—	34.91	—
Rock	33.10	—	26.01	—	35.46	—
Trunk	28.61	—	23.79	—	29.21	—
All	54.45	—	47.67	—	56.03	—
Semantic segmentation branch						
Water	—	94.00	—	93.60	—	94.33
Blooms	—	89.71	—	89.09	—	90.76
Grass	—	53.81	—	52.00	—	54.91
Dirt	—	40.10	—	39.80	—	40.50
Tree	—	86.51	—	85.91	—	85.82
House	—	47.91	—	46.71	—	47.84
Mountain	—	41.50	—	40.90	—	43.81
Sky	—	94.10	—	93.61	—	94.00
All	—	68.46	—	67.70	—	69.00

localization of textures from different areas, while the CBAM module exploits the inter-channel and inter-spatial relationships of features, focusing attention on salient properties of textures. Also, the PAN module plays a critical role in merging information from all the enhanced feature maps at different scales and resolutions derived from textures of different sizes.

Finally, it is also worth mentioning some limitations detected in scenes with different complexities. Fig. 17 shows some examples with the suboptimal result for both, object detection and semantic

segmentation. In the first row, the object detection performance is evidenced, where the regions highlighted by red circles reveal the incorrect localization and identification of some navigation obstacles, such as trunk and rock. It is postulated that this failure can be attributed to two possible reasons: a) the low diversity of these classes in the training dataset, and b) class imbalance, where trunk and rock classes are at a disadvantage compared to other classes, evidencing their minority status within the training dataset, as shown in Fig. 12. On the other hand, the second row presents the semantic segmentation result, where some mis-segmented areas are identified, also highlighted by red circles. For example, the misclassification of certain pixels belonging to the house class as tree is observed. The possible reasons may be attributable, once again, to the lack of diversity in the data and class imbalance, as evidenced in Fig. 13.

5. Conclusions

In this study, we propose a dual-task model based on CNNs that integrates object detection and semantic segmentation in lentic water scenes to extract all available global information by simultaneously identifying and locating navigational obstacles and amorphous semantic structures such as water bodies and cyanobacterial blooms through a MVS onboard a USV, i.e. with limited processing capabilities. In this regard, the overall architecture is inspired by YOLOv4 and YOLOv5, known for their state-of-the-art real-time results on conventional GPUs. The neural architectures for object detection and semantic segmentation share the backbone and the neck but differ in their heads.

The Spatial Pyramid Pooling (SPP) mechanism has been integrated into the proposed architecture to facilitate the detection of objects of different sizes as well as textures of different areas, and the Convolutional Block Attention Module (CBAM) to focus attention on difficult regions in adverse environmental conditions. Experimental results



Fig. 17. Visual presentation of the suboptimal result for both object detection and semantic segmentation. The first row corresponds to the object detection result and the second row to the semantic segmentation result.

evidence that the integration of the mechanisms is highly effective, as they guide the model to focus on the target objects and textures detection, resulting in a significant improvement in overall model performance.

We prepared two datasets specially adapted to the characteristics of lentic water scenes: one for the object detection task and one for the semantic segmentation task. For the annotation of the images in these sets, we used two state-of-the-art models: Grounding DINO and Segment Anything Model (SAM). Subsequently, we refined these annotations manually. Given the interest in detecting cyanobacterial blooms in water bodies, we combine our dataset with RS-CyanoHABs, which contains images of lakes and reservoirs with real and synthetic patches of cyanobacteria, captured from the perspective of an Autonomous Surface Vehicle (ASV). It is obvious that the more images and the better balanced the dataset is with respect to classes and objects, the better it will perform, being necessary in the future to move towards the best possible balancing.

We train and evaluate various configurations of the proposed model, keeping the same backbone (CSPDarknet-53) and the same detection head (YOLOv3 head) in the object detection branch while varying the segmentation head (BiSeNet, DeepLabV3+, and PSPNet) in the semantic segmentation branch. The training process is jointly performed end-to-end using object detection and semantic segmentation datasets. In addition, the model benefits from the pre-trained weights of the yolov5s model.

Overall, our dual-task model approach, together with the inclusion of the SPP and CBAM mechanisms, has shown great potential to address specific challenges present in environments such as lentic water scenes.

Furthermore, the proposed model can be generalized to other outdoor scenarios with the presence of adverse phenomena (variability in illumination, textures with whimsical and amorphous forms, objects of different sizes, reflections in aquatic environments, among others), where there are simultaneously objects to detect and amorphous textures. Examples are the detection of spills on the water surface (oil, petroleum products, fertilizers) or the detection of waste by using MVS, both inland and coastal.

Nevertheless, due to the instability of the MVSs on board the USVs a limitation of the proposed approach comes from the fact that temporal information, coming from image sequences (videos), to detect salient objects [74] or textures is not applicable.

CRedit authorship contribution statement

Fredy Barrientos-Espilco: Writing – review & editing, Writing – original draft, Visualization, Resources, Methodology, Investigation. **Gonzalo Pajares:** Writing – review & editing, Validation, Formal analysis. **María J. Gómez-Silva:** Writing – review & editing, Validation, Formal analysis. **Eva Besada-Portas:** Writing – review & editing, Validation, Funding acquisition, Formal analysis.

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

The authors are unable or have chosen not to specify which data has been used.

Acknowledgment

This work has been supported by the Research Projects IA-GES-BLOOM-CM (Y2020/TCS-6420) of the Synergic program of the Comunidad Autónoma de Madrid, SMART-BLOOMS (TED2021-130123B-

I00), funded by MCIN/AEI/10.13039/501100011033 and the European Union NextGenerationEU/PRTR, and INSERTION (PID2021-127648OB-C33) of the Knowledge Generation Projects program of the Spanish Ministry of Science and Innovation. The first author, Fredy Barrientos-Espilco, is supported by a scholarship by PRONABEC, Ministry of Education of Peru. The authors thank the anonymous referees for their very valuable comments.

References

- [1] M.V. Storey, B. van der Gaag, B.P. Burns, Advances in on-line drinking water quality monitoring and early warning systems, *Water Res* 45 (2011) 741–747, <https://doi.org/10.1016/j.watres.2010.08.049>.
- [2] M. Dunbabin, A. Grinham, Quantifying Spatiotemporal Greenhouse Gas Emissions Using Autonomous Surface Vehicles, *J. Field Robot.* 34 (2017) 151–169, <https://doi.org/10.1002/rob.21665>.
- [3] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection, in: 2016 IEEE Conf. Comput. Vis. Pattern Recognit, CVPR, 2016, pp. 779–788, <https://doi.org/10.1109/CVPR.2016.91>.
- [4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, S.S. D. Single Shot MultiBox, Detector, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), *Comput. Vis. – ECCV 2016*, Springer International Publishing, Cham, 2016, pp. 21–37, https://doi.org/10.1007/978-3-319-46448-0_2.
- [5] R. Girshick, Fast R-CNN, in: 2015 IEEE Int. Conf. Comput. Vis. ICCV, 2015: pp. 1440–1448, <https://doi.org/10.1109/ICCV.2015.169>.
- [6] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal Loss for Dense Object Detection, *IEEE Trans. Pattern Anal. Mach. Intell.* 42 (2020) 318–327, <https://doi.org/10.1109/TPAMI.2018.2858826>.
- [7] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: 2015 IEEE Conf. Comput. Vis. Pattern Recognit. CVPR, 2015: pp. 3431–3440, <https://doi.org/10.1109/CVPR.2015.7298965>.
- [8] O. Ronneberger, P. Fischer, T. Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, in: N. Navab, J. Hornegger, W.M. Wells, A.F. Frangi (Eds.), *Med. Image Comput. Comput.-Assist. Interv. – MICCAI 2015*, Springer International Publishing, Cham, 2015, pp. 234–241, https://doi.org/10.1007/978-3-319-24574-4_28.
- [9] V. Badrinarayanan, A. Kendall, R. Cipolla, SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, *IEEE Trans. Pattern Anal. Mach. Intell.* 39 (2017) 2481–2495, <https://doi.org/10.1109/TPAMI.2016.2644615>.
- [10] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, Pyramid Scene Parsing Network, in: 2017 IEEE Conf. Comput. Vis. Pattern Recognit. CVPR, 2017: pp. 6230–6239, <https://doi.org/10.1109/CVPR.2017.660>.
- [11] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs, *IEEE Trans. Pattern Anal. Mach. Intell.* 40 (2018) 834–848, <https://doi.org/10.1109/TPAMI.2017.2699184>.
- [12] W. Zhang, K. Wang, Y. Wang, L. Yan, F.-Y. Wang, A loss-balanced multi-task model for simultaneous detection and segmentation, *Neurocomputing* 428 (2021) 65–78, <https://doi.org/10.1016/j.neucom.2020.11.024>.
- [13] S. Zhang, Z. Zhang, L. Sun, W. Qin, One For All: A Mutual Enhancement Method for Object Detection and Semantic Segmentation, *Appl. Sci.* 10 (2020) 13, <https://doi.org/10.3390/app10010013>.
- [14] Z. Nan, J. Peng, J. Jiang, H. Chen, B. Yang, J. Xin, N. Zheng, A joint object detection and semantic segmentation model with cross-attention and inner-attention mechanisms, *Neurocomputing* 463 (2021) 212–225, <https://doi.org/10.1016/j.neucom.2021.08.031>.
- [15] J. Peng, Z. Nan, L. Xu, J. Xin, N. Zheng, A Deep Model for Joint Object Detection and Semantic Segmentation in Traffic Scenes, *Int. Jt. Conf. Neural Netw. IJCNN* 2020 (2020) 1–8, <https://doi.org/10.1109/IJCNN48605.2020.9206883>.
- [16] L. Weng, Y. Wang, F. Gao, Traffic Scene Perception Based on Joint Object Detection and Semantic Segmentation, *Neural Process. Lett.* (2022), <https://doi.org/10.1007/s11063-022-10864-z>.
- [17] L. Chen, Z. Yang, J. Ma, Z. Luo, Driving Scene Perception Network: Real-time Joint Detection, Depth Estimation and Semantic Segmentation, in: 2018 IEEE Winter Conf. Appl. Comput. Vis. WACV, 2018: pp. 1283–1291, <https://doi.org/10.1109/WACV.2018.00145>.
- [18] N.O. Salscheider, Simultaneous Object Detection and Semantic Segmentation, (2020). (<http://arxiv.org/abs/1905.02285>) (accessed March 6, 2024).
- [19] B. Li, S. Liu, W. Xu, W. Qiu, Real-time object detection and semantic segmentation for autonomous driving. in: MIPPR 2017 Autom. Target Recognit. Navig., SPIE, 2018, pp. 167–174, <https://doi.org/10.1117/12.2288713>.
- [20] A. Bochkovskiy, C.-Y. Wang, H.-Y.M. Liao, YOLOv4: Optim. Speed Accuracy Object Detect. (2020), <https://doi.org/10.48550/arXiv.2004.10934>.
- [21] X. Chen, Y. Liu, K. Achuthan, WODIS: Water Obstacle Detection Network Based on Image Segmentation for Autonomous Surface Vehicles in Maritime Environments, *IEEE Trans. Instrum. Meas.* 70 (2021) 1–13, <https://doi.org/10.1109/TIM.2021.3092070>.
- [22] F. Chollet, Xception: Deep Learning with Depthwise Separable Convolutions, in: 2017 IEEE Conf. Comput. Vis. Pattern Recognit. CVPR, 2017: pp. 1800–1807, <https://doi.org/10.1109/CVPR.2017.195>.
- [23] B. Bovcon, M. Kristan, WaSR–A Water Segmentation and Refinement Maritime Obstacle Detection Network, *IEEE Trans. Cybern.* (2021) 1–14, <https://doi.org/10.1109/TCYB.2021.3085856>.

- [24] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, in: IEEE Computer Society, 2016: pp. 770–778. <https://doi.org/10.1109/CVPR.2016.90>.
- [25] T. Cane, J. Ferryman, Evaluating deep semantic segmentation networks for object detection in maritime surveillance, in: 2018 15th IEEE Int. Conf. Adv. Video Signal Based Surveill. AVSS, 2018: pp. 1–6. <https://doi.org/10.1109/AVSS.2018.8639077>.
- [26] J. Taipalmaa, N. Passalis, H. Zhang, M. Gabbouj, J. Raitoharju, High-Resolution Water Segmentation for Autonomous Unmanned Surface Vehicles: a Novel Dataset and Evaluation, IEEE 29th Int. Workshop Mach. Learn. Signal Process. MLSP, 2019 (2019) 1–6. <https://doi.org/10.1109/MLSP.2019.8918694>.
- [27] M.A.M. Adam, A.I. Ibrahim, Z.Z. Abidin, H.F.M. Zaki, Deep Learning-Based Water Segmentation for Autonomous Surface Vessel, IOP Conf. Ser. Earth Environ. Sci. 540 (2020) 012055, <https://doi.org/10.1088/1755-1315/540/1/012055>.
- [28] L. Steccanella, D. Bloisi, J. Blum, A. Farinelli, Deep Learning Waterline Detection for Low-Cost Autonomous Boats, 15, in: M. Strand, R. Dillmann, E. Menegatti, S. Ghidoni (Eds.), Intell. Auton. Syst, Springer International Publishing, Cham, 2019, pp. 613–625, https://doi.org/10.1007/978-3-030-01370-7_48, 15.
- [29] W. Zhan, C. Xiao, Y. Wen, C. Zhou, H. Yuan, S. Xiu, Y. Zhang, X. Zou, X. Liu, Q. Li, Autonomous Visual Perception for Unmanned Surface Vehicle Navigation in an Unknown Environment, Sensors 19 (2019) 2216, <https://doi.org/10.3390/s19102216>.
- [30] T.S. Akiyama, J. Marcato Junior, W.N. Gonçalves, P.O. Bressan, A. Eltner, F. Binder, T. Singer, DEEP LEARNING APPLIED TO WATER SEGMENTATION, : Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci., Copernic. GmbH (2020) 1189–1193, <https://doi.org/10.5194/isprs-archives-XLIII-B2-2020-1189-2020>.
- [31] R. Vandaele, S.L. Dance, V. Ojha, Automated Water Segmentation and River Level Detection on Camera Images Using Transfer Learning, in: Z. Akata, A. Geiger, T. Sattler (Eds.), Pattern Recognit, Springer International Publishing, Cham, 2021, pp. 232–245, https://doi.org/10.1007/978-3-030-71278-5_17.
- [32] L. Weng, Y. Xu, M. Xia, Y. Zhang, J. Liu, Y. Xu, Water Areas Segmentation from Remote Sensing Images Using a Separable Residual SegNet Network, ISPRS Int. J. Geo-Inf. 9 (2020) 256, <https://doi.org/10.3390/ijgi9040256>.
- [33] F. Barrientos-Espillo, E. Gascó, C.I. López-González, M.J. Gómez-Silva, G. Pajares, Semantic segmentation based on Deep learning for the detection of Cyanobacterial Harmful Algal Blooms (CyanoHABs) using synthetic images, Appl. Soft Comput. 141 (2023) 110315, <https://doi.org/10.1016/j.asoc.2023.110315>.
- [34] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, L.-H. Yeh, CSPNet: A New Backbone that can Enhance Learning Capability of CNN, in: 2020 IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshop CVPRW, 2020: pp. 1571–1580. <https://doi.org/10.1109/CVPRW50498.2020.00203>.
- [35] K. He, X. Zhang, S. Ren, J. Sun, Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, IEEE Trans. Pattern Anal. Mach. Intell. 37 (2015) 1904–1916, <https://doi.org/10.1109/TPAMI.2015.2389824>.
- [36] S. Liu, L. Qi, H. Qin, J. Shi, J. Jia, Path Aggregation Network for Instance Segmentation, in: 2018 IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2018: pp. 8759–8768. <https://doi.org/10.1109/CVPR.2018.00913>.
- [37] S. Woo, J. Park, J.-Y. Lee, I.S. Kweon, CBAM: Convolutional Block Attention Module, in: V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss (Eds.), Comput. Vis. – ECCV 2018, Springer International Publishing, Cham, 2018, pp. 3–19, https://doi.org/10.1007/978-3-030-01234-2_1.
- [38] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, N. Sang, BiSeNet: Bilateral Segmentation Network for Real-Time Semantic Segmentation, in: V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss (Eds.), Comput. Vis. – ECCV 2018, Springer International Publishing, Cham, 2018, pp. 334–349, https://doi.org/10.1007/978-3-030-01261-8_20.
- [39] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation, in: V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss (Eds.), Comput. Vis. – ECCV 2018, Springer International Publishing, Cham, 2018, pp. 833–851, https://doi.org/10.1007/978-3-030-01234-2_49.
- [40] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, IEEE Conf. Comput. Vis. Pattern Recognit. 2014 (2014) 580–587, <https://doi.org/10.1109/CVPR.2014.81>.
- [41] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, IEEE Trans. Pattern Anal. Mach. Intell. 39 (2017) 1137–1149, <https://doi.org/10.1109/TPAMI.2016.2577031>.
- [42] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, IEEE Trans. Pattern Anal. Mach. Intell. 42 (2020) 386–397, <https://doi.org/10.1109/TPAMI.2018.2844175>.
- [43] J. Redmon, A. Farhadi, YOLO9000: Better, Faster, Stronger, in: 2017 IEEE Conf. Comput. Vis. Pattern Recognit. CVPR, 2017: pp. 6517–6525. <https://doi.org/10.1109/CVPR.2017.690>.
- [44] J. Redmon, A. Farhadi, YOLOv3: An Incremental Improvement, (2018). <https://arxiv.org/abs/1804.02767>.
- [45] S. Luo, J. Yu, Y. Xi, X. Liao, Aircraft Target Detection in Remote Sensing Images Based on Improved YOLOv5, IEEE Access 10 (2022) 5184–5192, <https://doi.org/10.1109/ACCESS.2022.3140876>.
- [46] M. Teichmann, M. Weber, M. Zöllner, R. Cipolla, R. Urtasun, MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving, in: 2018 IEEE Intell. Veh. Symp. IV, 2018: pp. 1013–1020. <https://doi.org/10.1109/IVS.2018.8500504>.
- [47] M. Tan, Q. Le, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, in: Proc. 36th Int. Conf. Mach. Learn., PMLR, 2019: pp. 6105–6114. <https://proceedings.mlr.press/v97/tan19a.html> (accessed March 6, 2024).
- [48] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature Pyramid Networks for Object Detection, in: 2017 IEEE Conf. Comput. Vis. Pattern Recognit. CVPR, 2017: pp. 936–944. <https://doi.org/10.1109/CVPR.2017.106>.
- [49] J. Hu, L. Shen, S. Albanie, G. Sun, E. Wu, Squeeze-and-Excitation Networks, IEEE Trans. Pattern Anal. Mach. Intell. 42 (2020) 2011–2023, <https://doi.org/10.1109/TPAMI.2019.2913372>.
- [50] M.D. Zeiler, R. Fergus, Visualizing and Understanding Convolutional Networks, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Comput. Vis. – ECCV 2014, Springer International Publishing, Cham, 2014, pp. 818–833, https://doi.org/10.1007/978-3-319-10590-1_53.
- [51] S. Zagoruyko, N. Komodakis, Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer, in: 2022. <https://openreview.net/forum?id=Sks9ajex> (accessed March 6, 2024).
- [52] A. Zafar, M. Aamir, N. Mohd Nawi, A. Arshad, S. Riaz, A. Alruban, A.K. Dutta, S. Almotairi, A Comparison of Pooling Methods for Convolutional Neural Networks. Applied Sciences 12, 17, 8643. <https://doi.org/10.3390/app12178643>.
- [53] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, D. Ren, Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression, Proc. AAAI Conf. Artif. Intell. 34 (2020) 12993–13000, <https://doi.org/10.1609/aaai.v34i07.6999>.
- [54] L.-C. Chen, G. Papandreou, F. Schroff, H. Adam, Rethinking Atrous Convolution for Semantic Image Segmentation, ArXiv170605587 Cs. (2017). <http://arxiv.org/abs/1706.05587> (accessed March 6, 2024).
- [55] B. Bovcon, J. Muhović, J. Perš, M. Kristan, The MaStr1325 dataset for training deep USV obstacle detection models, in: 2019 IEEE/RSJ Int. Conf. Intell. Robots Syst. IROS, 2019: pp. 3431–3438. <https://doi.org/10.1109/IROS40897.2019.8967909>.
- [56] B. Bovcon, R. Mandeljc, J. Perš, M. Kristan, Stereo obstacle detection for unmanned surface vehicles by IMU-assisted semantic segmentation, Robot. Auton. Syst. 104 (2018) 1–13, <https://doi.org/10.1016/j.robot.2018.02.017>.
- [57] J. Liu, H. Li, J. Luo, S. Xie, Y. Sun, Efficient obstacle detection based on prior estimation network and spatially constrained mixture model for unmanned surface vehicles, J. Field Robot. 38 (2021) 212–228, <https://doi.org/10.1002/rob.21983>.
- [58] D.K. Prasad, D. Rajan, L. Rachmawati, E. Rajabally, C. Quek, Video Processing From Electro-Optical Sensors for Object Detection and Tracking in a Maritime Environment: A Survey, IEEE Trans. Intell. Transp. Syst. 18 (2017) 1993–2016, <https://doi.org/10.1109/TITS.2016.2634580>.
- [59] Y. Cheng, M. Jiang, J. Zhu, Y. Liu, Are We Ready for Unmanned Surface Vehicles in Inland Waterways? The USVINland Multisensor Dataset and Benchmark, IEEE Robot. Autom. Lett. 6 (2021) 3964–3970, <https://doi.org/10.1109/LRA.2021.3067271>.
- [60] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, T. Aila, Training Generative Adversarial Networks with Limited Data, in: Adv. Neural Inf. Process. Syst., Curran Associates, Inc., 2020: pp. 12104–12114. <https://proceedings.neurips.cc/paper/2020/hash/8d30aa96e72440759f74bd2306c1fa3d-Abstract.html> (accessed March 6, 2024).
- [61] L. Gatys, A. Ecker, M. Bethge, A Neural Algorithm of Artistic Style, J. Vis. 16 (2016) 326, <https://doi.org/10.1167/16.12.326>.
- [62] Flickr, (n.d.). <https://www.flickr.com/about> (accessed March 6, 2024).
- [63] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, L. Zhang, Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection, (2023). <https://doi.org/10.48550/arXiv.2303.05499>.
- [64] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A.C. Berg, W.-Y. Lo, P. Dollár, R. Girshick, Segm. Anything (2023), <https://doi.org/10.48550/arXiv.2304.02643>.
- [65] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L.M. Ni, H.-Y. Shum, DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection, (2022). <https://doi.org/10.48550/arXiv.2203.03605>.
- [66] L.H. Li, P. Zhang, H. Zhang, J. Yang, C. Li, Y. Zhong, L. Wang, L. Yuan, L. Zhang, J.-N. Hwang, K.-W. Chang, J. Gao, Ground Lang. -Image Pre-Train. (2022), <https://doi.org/10.48550/arXiv.2112.03857>.
- [67] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the Inception Architecture for Computer Vision, in: 2016 IEEE Conf. Comput. Vis. Pattern Recognit. CVPR, 2016: pp. 2818–2826. <https://doi.org/10.1109/CVPR.2016.308>.
- [68] L. Bottou, Large-Scale Machine Learning with Stochastic Gradient Descent, in: Y. Lechevallier, G. Saporta (Eds.), Proc. COMPSTAT2010, Physica-Verlag HD, Heidelberg, 2010, pp. 177–186, https://doi.org/10.1007/978-3-7908-2604-3_16.
- [69] I. Goodfellow, Y. Bengio, A. Courville, A., Deep learning, MIT Press, 2016. <https://www.deeplearningbook.org/>.
- [70] L.N. Smith, N. Topin, Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates, (2018). [arXiv:1708.07120 \[cs.LG\]](https://arxiv.org/abs/1708.07120).
- [71] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft COCO: Common Objects in Context, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Comput. Vis. – ECCV 2014, Springer International Publishing, Cham, 2014, pp. 740–755, https://doi.org/10.1007/978-3-319-10602-1_48.
- [72] D. Misra, Mish: A Self Regularized Non-Monotonic Activation Function, (2020). <https://arxiv.org/abs/1808.08681>.
- [73] G. Ghiasi, T.-Y. Lin, Q.V. Le, DropBlock: A regularization method for convolutional networks, (2018). <https://doi.org/10.48550/arXiv.1810.12890>.
- [74] C. Xu, Z. Gao, H. Zhang, S. Li, V.H.C. de Albuquerque, Video salient object detection using dual-stream spatiotemporal attention, Appl. Soft Comput. 108 (2021) 107433, <https://doi.org/10.1016/j.asoc.2021.107433>.