TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 24, No. 5, October 2026, pp. 1561∼1580
ISSN: 1693-6930, DOI: 10.12928/TELKOMNIKA.v24i5.27678

NiteTron: a novel nighttime bunaken sea turtle species detection

Muhamad Dwisnanto Putro[1], Jane Litouw[2], Abdul Haris Ontowirjo[2], Sherwin Reinaldo U. Sompie[2], Hebron Prasetya[1]
[1]Master Program of Informatics, Postgraduate Program, Sam Ratulangi University, Manado, Indonesia
[2]Department of Electrical Engineering, Sam Ratulangi University, Manado, Indonesia

Article Info

Article history:

Received Nov 19, 2025
Revised Jun 2, 2026
Accepted Jun 30, 2026

Keywords:

Bunaken
Computer vision
Deep learning
Nighttime
Sea turtle
Underwater detection

ABSTRACT

Automated monitoring is essential for sea turtle conservation, and vision-based algorithms offer a non-intrusive approach to observing turtles without disturbing their natural habitat. However, nighttime conditions introduce significant challenges, as low illumination severely reduces the visibility of discriminative features. This limitation requires robust feature extraction to capture the limited important features. Recent advances in computer vision, such as you only look once (YOLO) 11-nano, enable real-time detection and allow optimization under challenging conditions. In this study, we propose NiteTron, a nighttime sea turtle species detector that uses a lightweight architecture with enhanced feature refinement. The network modifies YOLO11-nano to create a more efficient variant, YOLO11-pico. To improve detection in low-light scenarios, we introduce a strong feature modulation (SFM) module. This module enhances attention to relevant features by expanding the receptive field in the final backbone stage. We also incorporate a sparse separated convolution (S2C) module to extract key information efficiently while balancing accuracy and computational cost. The proposed model achieves high precision with fewer parameters and lower giga floating-point operations per second (GFLOPs) than YOLO11-nano. Experiments demonstrate superior performance, achieving 0.682 mean average precision (mAP)@0.5:0.95 for nighttime sea turtle detection. Inference tests also confirm efficient central processing unit (CPU) deployment, reaching 21.18 frames per second (FPS) on a desktop and 2.65 FPS on an edge device.

Corresponding Author:

Muhamad Dwisnanto Putro
Master Program of Informatics, Postgraduate Program, Sam Ratulangi University
Manado, Indonesia
Email: dwisnantoputro@unsrat.ac.id

1. INTRODUCTION

Sea turtles play a crucial role in maintaining the stability of marine ecosystems. Notably, Bunaken Marine Park provides a natural habitat for three distinct sea turtle species and is recognized as a marine conservation area with high biodiversity. The three species commonly found in Bunaken are the green, olive ridley, and hawksbill Turtle. Each of these species makes a unique contribution to the marine ecosystem. Therefore, awareness of their location is crucial for protecting their habitat and population. To address this issue, continuous monitoring through automated detection systems powered by computer vision is essential to prevent extinction and ensure long-term survival. These systems enable around-the-clock monitoring while minimizing

resource and operational costs. A previous study developed a you only look once (YOLO)v7-based detection model combined with the bag of tricks-sort (BoT-SORT) algorithm for automatic counting and tracking of green turtles in Japanese coastal areas [1]. The approach achieved promising results, with the best performance reaching a mean average precision (mAP@50:95) of 0.620. However, since the method relied on aerial drone views, its effectiveness was limited when detecting underwater turtles due to turbidity, surface ripples, and reduced visibility. On the other hand, the studies in [2], [3] have successfully implemented underwater sea turtle detection as a foundational step toward automated conservation systems. However, continuous monitoring faces significant challenges in nighttime environments due to low illumination. While previous studies on underwater turtle detection have shown promising results, they are limited to daytime conditions, reducing their applicability in low-light scenarios. Low light conditions reduce key features used to identify turtle species, such as shell patterns and coloration. Moreover, underwater environments introduce additional challenges, such as motion blur and visual distortion, which further compromise recognition accuracy. In order to address these challenges, this research develops a model that performs accurately and efficiently in nighttime conditions, making it suitable for deployment on low-cost devices. The proposed model integrates deep learning methods and ensures reliable detection and recognition of turtle species in low-light environments. This approach not only improves accuracy but also reduces reliance on human intervention, minimizing errors and enhancing operational efficiency. Moreover, it enables continuous, real-time monitoring in challenging underwater settings, supporting scalable and sustainable marine conservation efforts.

Convolutional neural networks (CNNs) are widely used in object detection [4], [5]. One of the most well-known object detectors is the YOLO [6], which relies on CNNs as its core component and is capable of real-time operation. The demand for lightweight and efficient algorithms in computer vision led to the development of YOLO11-nano [7]. These networks utilize fewer parameters and giga floating-point operations per second (GFLOPs) to operate. However, it delivers lower precision compared to other YOLO11 variants. Hence, the performance needs to be enhanced to improve its ability to handle nighttime and underwater challenges. Its lightweight and efficient design demonstrates strong potential for further optimization under such demanding conditions. Although improving performance often demands higher computational costs, several studies have introduced lightweight modules that boost recognition accuracy without significantly increasing complexity [8]-[10].

This paper proposes a novel nighttime underwater sea turtle detector with vision approaches, named NiteTron. Based on an efficient modification of YOLO11-nano, called YOLO11-pico. The model achieves competitive performance compared to other recent YOLO architectures, which utilize lightweight convolutional modules to improve detection accuracy in low-light environments. The network introduces two main modules to enhance the backbone. Strong feature modulation (SFM) captures a larger receptive field using a big kernel and applies a gate operation to select relevant cells. It also fuses information and strengthens feature relationships. Furthermore, sparse separated convolution (S2C) efficiently extracts characteristic features of turtles by combining partial and depthwise convolutions. It also integrates enhancing channel (EC) modules, which use broadcast multiplication to emphasize relevant feature channels. The main contributions of this work are summarized as follows:

- A novel sea turtle detection method is proposed for nighttime conditions, utilizing an efficient deep learning approach that optimizes both detection performance and inference speed.
- A new efficient variant of the YOLO11 model is introduced, utilizing a series of lightweight operations and a reduced layer structure to effectively extract discriminative features while minimizing trainable parameters and computational complexity.
- A new efficient feature extraction block is proposed, employing lightweight convolutional layers. This module partitions the input feature map and integrates a channel enhancement mechanism at each layer, effectively reducing the number of trainable parameters and computational cost.
- A novel enhancement modulation module is offered, implementing sequential attention blocks that include channel attention and gated operations. This module enhances the feature extraction capability by capturing essential contextual information and reinforcing the relationships between spatial and global features. Its plug-and-play design encourages a robust and adaptable attention module suitable for integration into deep learning architecture.
- A comprehensive model analysis is conducted to evaluate the effectiveness and efficiency of the proposed lightweight and enhancement module. The study assesses both performance and computational overhead introduced by the additional blocks. Furthermore, the proposed detector is evaluated on nighttime sea turtle

imagery from Bunaken and demonstrates competitive performance in mAP compared to other lightweight YOLO-based detectors. To validate the efficiency of the proposed model, additional metrics are reported, including the number of parameters, GFLOPs, and runtime speed on different central processing unit (CPU) hardware platforms.

2. RELATED WORKS

Underwater object detection using CNN models has been extensively applied, with a primary focus on improving detection accuracy. Previous works, such as attention-free hybrid network (AFHNet) [11], YOLOv8-UC [12], HLAST-ACNet [13], and enhanced YOLOv5 [14]. They have demonstrated that CNN-based architectures can achieve real-time underwater object detection with improved performance. Li *et al.* [15], study proposed an underwater detector based on the real-time detection transformer (RT-DETR) architecture, called Underwater-DETR (UW-DETR), which outperformed several RT-DETR variants. However, it requires high computational resources and a large number of parameters. Additionally, YOLOv7-PE [16] improves the balance between accuracy and efficiency. It is built upon the YOLOv7 architecture, incorporating the convolutional block attention module (CBAM) module to enhance detection accuracy and the cross stage partial spatial pyramid pooling-fast (CSPSPPF) module to reduce computational cost. Nevertheless, its performance has not yet been evaluated under low-light conditions. Furthermore, Zhang *et al.* [17] introduced an efficient underwater object detector based on YOLOv8, with fewer parameters and lower GFLOPs compared to YOLOv8-Large. However, the model exhibits limitations in handling overlapping objects, which may result in missed detections of underwater objects.

Moreover, deep learning methods have significantly advanced marine animal detection for ecosystem monitoring. YOLOv7 was employed as an object detector within a weight prediction system to assist in turtle sorting [18]. Similarly, Zakry *et al.* [19] applied deep learning techniques to sea turtle detection, supporting conservation efforts through more efficient and real-time monitoring. Nevertheless, grayscale images used in model training pose challenges for deployment in real-time applications using red, green, and blue (RGB) cameras. The study in Baek *et al.* [20] conducts a comparative analysis using YOLOv5-seg for sea turtle classification with instance segmentation, highlighting its potential for precise object delineation, though it does not focus on model optimization or deployment efficiency. Another study, Liu *et al.* [21] proposes global attention mechanism and transfer learning (PLGAT), an underwater *Plectropomus leopardus* detector based on a modified YOLOv8. It achieves 0.958 mAP on the *Plectropomus leopardus recognition dataset* (PLRD) dataset and 0.974 on the Brackish dataset. However, it requires 7.4 million parameters, making it unsuitable for edge devices.

As key contributors to marine ecosystems, sea turtles have become a focus of research in population monitoring and conservation. Adam *et al.* [22] introduced a long-span dataset for sea turtle re-identification based on stable external features, enabling continuous individual tracking over time. However, the recognition system is not designed for video-based or real-time deployment. Additionally, previous work [2] proposed a modified YOLOv8-nano network for sea turtle detection, introducing the excited group attention (EGA) module to generate multi-scale representations across various receptive fields. The module employs channel shuffle operations to enhance feature blending. Using this approach, the proposed model achieved a mAP50:95 score of 0.616, representing a 0.049 improvement over the original YOLOv8-nano network. This model also reduces the number of channels compared to the original YOLOv8-nano, making it more lightweight and efficient. Another recent study, Putro *et al.* [3] presented an efficient sea turtle detector by limiting the maximum number of channels in the network. This modification to YOLOv8 reduces the parameter count by 1.1 million. Additionally, the authors propose dual-spatial enhancement dan gradual receptive enhancement modules to improve detection accuracy, achieving a mAP50:95 score of 0.604.

To enable continuous sea turtle monitoring, the proposed method must handle low-light conditions at night, which reduce distinguishing features. Several studies have explored nighttime object detection [23]-[25]. Zhao *et al.* [26] proposed a YOLOv5-based approach for nighttime pedestrian detection using infrared cameras to capture low-light environments, achieving a mAP score of 93.62. Nevertheless, the method is limited to infrared imagery. Further evaluation on RGB data is required to ensure more flexible and practical deployment. Additionally, Xu *et al.* [27] introduces a nighttime vehicle detection approach to support the development of intelligent transportation systems. The method enhances faster region-based convolutional neural network (R-CNN) by integrating deformable convolutional networks, achieving a mAP score of 83.3.

However, the evaluation is limited to a small-scale, single-scene dataset, which does not reflect the complexity and darker conditions often encountered in real-world scenarios.

Existing approaches exhibit limitations under low-light conditions, as reduced illumination degrades feature visibility and negatively impacts detection performance. Notably, Putro *et al.* [3] evaluates the model exclusively under daylight conditions, despite utilizing the same baseline dataset as this study. Therefore, this study proposes a nighttime underwater sea turtle detection model, named NiteTron, built on a modified YOLO11-nano architecture referred to as YOLO11-pico. The model is optimized for reliable performance in low-light conditions using standard RGB cameras, making it suitable for deployment on a wide range of devices. Additionally, the reduced number of channels makes the architecture both lightweight and efficient. Furthermore, two novel modules are incorporated to improve detection accuracy through lightweight enhancements while maintaining computational efficiency. The proposed method incorporates an efficient architecture that achieves competitive performance when compared to recent YOLO variants. In addition, this study focuses on the detection and recognition of sea turtles in Bunaken Marine Park, specifically targeting three species, namely green, olive ridley, and hawksbill.

3. PROPOSED ARCHITECTURE

This work proposes two novel modules in the main extractor, namely SFM and S2C, as part of the lightweight NiteTron architecture, resulting in an efficient nighttime underwater sea turtle detector. The S2C module is designed to extract distinguishing features through lightweight operations, while the SFM module captures a large receptive field. These modules enhance the network's performance while maintaining efficiency. This design enables the model to achieve real-time inference speed on promising devices.

3.1. Backbone

The backbone is a key component of YOLO11 architecture, responsible for extracting features from input images using convolutional layers and attention modules. As illustrated in Figure 1, it employs 3×3 convolutional blocks that function both as feature extractors and downsampling units when it uses a stride of 2 to reduce the spatial dimensions of the feature maps. Each convolutional layer is accompanied by batch normalization to stabilize data distribution across batches and channels, followed by the sigmoid linear unit (SiLU) activation function, which introduces non-linearity while retaining small negative values. Other than that, the C3K2 module plays a central role in feature extraction by providing a faster implementation of the cross stage partial (CSP) structure. It integrates the C3K module, which consists of three convolutional layers and includes an S2C unit to enhance feature quality. In addition, C3K is a module based on the bottleneck architecture, designed to support customizable convolutional kernel sizes. It splits the input feature maps along the channel dimension to enable efficient information extraction, using a 1×1 convolution to define the channel partitions. One-half of the channels passes through a bottleneck that replaces the standard convolution with the proposed S2C module. The output from the S2C module is fused using concatenate operation with the identity branch, which contains the remaining channels, using a concatenation operation. A final 1×1 convolution is applied to mix the concatenated feature maps and to capture channel relationships.

Furthermore, the spatial pyramid pooling faster (SPPF) module performs feature aggregation. It includes two 1×1 convolutions, positioned at the beginning and end of the process, along with three sequential average pooling operations using a 5×5 window in the middle of the process. However, the second 1×1 convolution is removed. The first 1×1 convolution is employed to set the number of channels before aggregation. Using multiple 5×5 pooling operations in sequence allows the model to capture increasingly larger receptive fields. For instance, two 5×5 layers approximate a 9×9 receptive field, while three layers approximate a 13 $\times$ 13 field. The resulting feature maps from each scale are combined to form a rich multi-scale representation using a concatenation operation. The SFM module is placed at the end of the backbone to enhance channel-wise information. Subsequently, a 1×1 convolution is applied at the end to mix the channel-wise information derived from the SFM output, further emphasizing the most relevant features for detection.

In order to improve efficiency, reducing the number of channels in several layers is essential for avoiding redundant feature maps and lowering computational cost. This reduction helps the map concentrate on meaningful object information. Among all YOLO versions, the nano variant is the smallest, as it utilizes fewer parameters and GFLOPs for operation. However, it still presents opportunities for further optimization, particularly for faster inference on CPU-based devices. This work proposes a new YOLO architecture, referred to as YOLO-pico, which adopts a more compact design by using fewer parameters and lower GFLOPs than

YOLO-nano. This compactness results from a deliberate reduction in the number of output channels across layers. The complete channel configuration is shown in Figure 1. The results demonstrate that increasing the number of channels does not necessarily lead to better accuracy. Instead, selectively retaining only essential channels yields a more efficient trade-off between accuracy and computational performance.

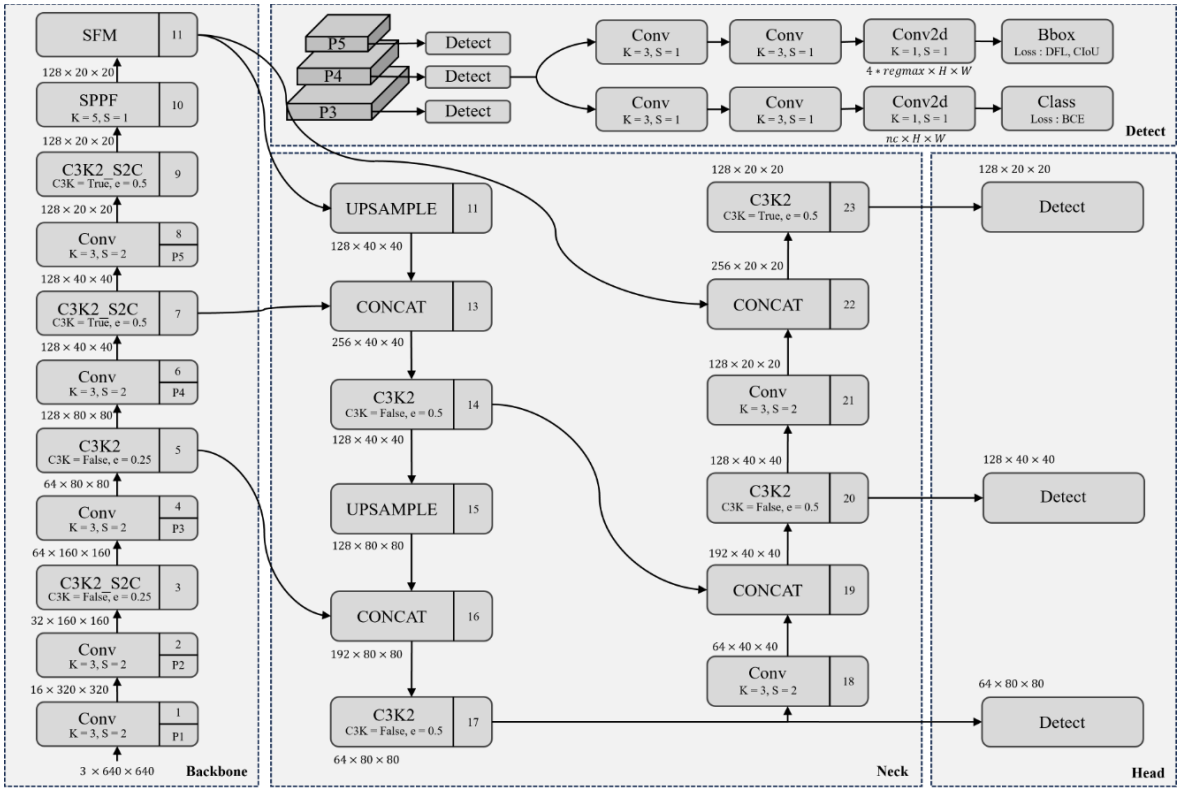


Figure 1. The proposed NiteTron architecture is based on YOLO11-pico and integrates two enhancement modules into the backbone. It emphasizes efficient feature extraction and channel refinement to improve detection performance

3.2. S2C

A backbone relies on convolutional operations to extract object information from input images, capturing local spatial features from the resulting feature maps. The original C3K2 module incorporates C3K (three convolutions with adaptive kernels), which includes a bottleneck structure that applies a 3 × 3 kernel twice, effectively approximating the receptive field of a 5 × 5 kernel. However, identifying which feature maps contain meaningful object information requires global context across channel representations, which typically demands a large number of parameters to model long-range dependencies. In this context, S2C aims to efficiently capture local spatial features by applying two types of convolution in separate processes with global representation. This mechanism relates spatial-level data without increasing the use of parameters and computational resources. To highlight essential feature maps, the captured global representation is used as a scaling factor for the corresponding feature maps, allowing them to be adjusted based on their global values. It is implemented at the medium and high-level feature stage to ensure that the most relevant features are enhanced while suppressing less information. As shown in Figure 2, the S2C module integrates standard convolution and depthwise convolution, which applies filters across all input channels and processes each half of the channel independently. The operation can be formulated as (1):

$$S2C(X) = Cv(EC(Concat[Cv_{s1}(X), Cv_{s2}(X)])) \tag{1}$$

The input feature maps X are split into two halves. Using only standard convolution to extract features would increase the number of parameters and computational cost. To address this, $S2C$ applies different

convolution types to each half of the channels. A standard convolution ($Cv_{s1}(X)$) is applied to one half to preserve inter-channel relationships, while a depthwise convolution ($Cv_{s2}(X)$) is utilized to the other half to enhance computational efficiency. The overall convolution process is illustrated as (2):

$$Cv_{s1} = Cv_{3\times3}(X_{s1}), Cv_{s2} = Dw_{3\times3}(X_{s2}) \tag{2}$$

To preserve local spatial relationships, a 3 × 3 kernel is employed, striking a balance between efficiency and neighborhood feature representation. Half of the channels are processed using $Cv_{3\times3}$, which consists of a 2D standard convolution with 3 × 3 kernel size followed by batch normalization to stabilize feature distribution, and a SiLU activation function to introduce non-linearity. While standard convolution captures inter-channel dependencies, the remaining half of the channels are processed using depthwise convolution with the same 3 × 3 kernel, also followed by batch normalization and SiLU activation. The two branches, X_{s1} and X_{s2}, represent the split feature maps obtained from the output of the convolution operation Cv, which utilizes a 1 × 1 kernel along with batch normalization and SiLU activation, denoted as (3).

$$[X_{s1}, X_{s2}] = Cv \tag{3}$$

This splitting design improves computational efficiency while preserving accurate feature extraction. Each branch follows a distinct processing strategy. Furthermore, the feature maps generated from different convolution branches are concatenated along the channel dimension to form a unified tensor, allowing features from different processing branches to be jointly processed. This fusion enhances the ability of feature extraction by capturing relationships between feature maps. To strengthen the interaction between outputs from standard and depthwise convolutions, a convolutional layer (Cv) is applied.

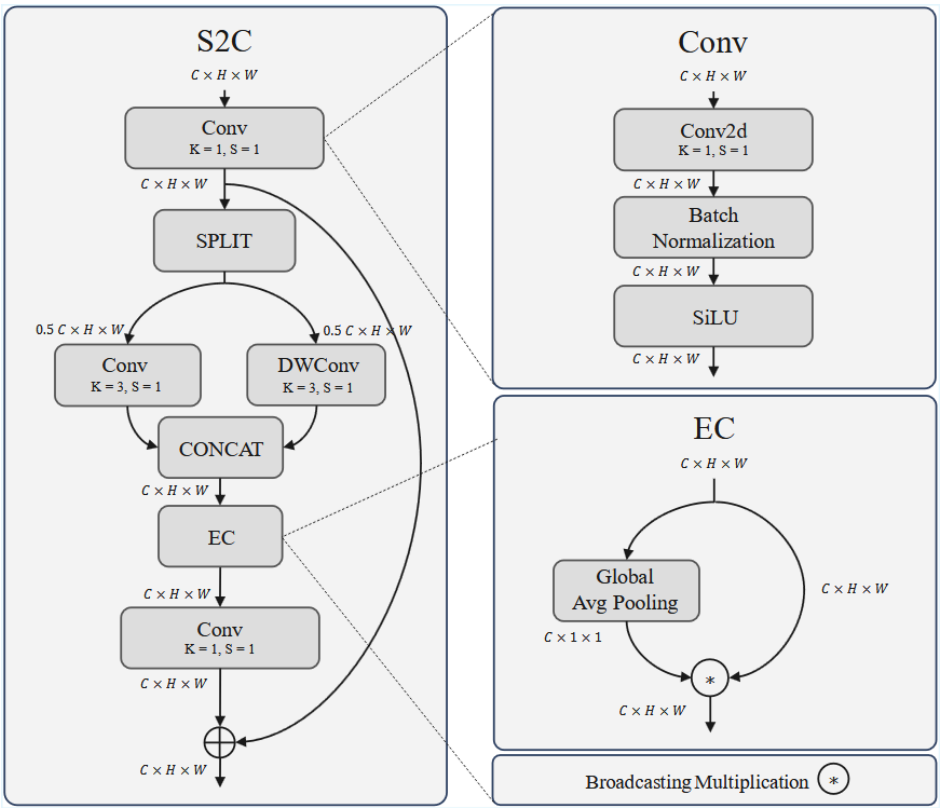


Figure 2. S2C, this block applies separate convolutions to obtain different feature representations, which are enhanced with attention channels

This layer consists of a 1 × 1 convolution, followed by batch normalization and a SiLU activation function. It focuses on EC-wise relationships, making the operation efficient for this task. Additionally, this

step restores the output to the intended number of channels, ensuring compatibility with the next stages of the network. In addition, improving the recognition of informative features requires identifying which channels carry the most significant information. Channel attention mechanisms commonly leverage global pooling to summarize each feature map into a representative value, which acts as a scaling factor for enhancing relevant channels. In the $S2C$ module, global average pooling (GAP) is used to perform this operation, preserving the original distribution of feature values. This process is defined as (4).

$$EC(X) = GAP(X) * X \tag{4}$$

The scaling operation improves feature map quality by emphasizing channels that carry more relevant object information. GAP extracts representative values from each channel, which serve as scaling factors for the input feature maps. This process represents EC that employs broadcasting multiplication via scaling values uniformly across the corresponding channels.

3.3. SFM

Attention mechanisms are effective techniques for enhancing model accuracy by enabling neural networks to focus on the most informative parts of an input. These modules refine feature representations by emphasizing specific feature maps that are most relevant to the object of interest. However, many existing attention mechanisms operate within a single domain, either channel-wise or spatially, which limits their effectiveness in complex or low-visibility scenarios. Channel attention modules capture global context by modeling the relationships across channels. While this helps the model prioritize semantically rich feature maps, it often misses spatial detail that is crucial for detecting small or fine-grained object features. On the other hand, spatial attention modules focus on identifying important regions within the feature map but tend to be computationally intensive, especially when capturing interactions across channels or operating on high-resolution data.

To improve detection accuracy in challenging scenarios, it is essential to incorporate both channel and spatial attention mechanisms. At the same time, maintaining low computational cost is critical to preserving overall model efficiency. In this work, we introduce a lightweight dual-attention module called SFM, which adopts the star technique [28] to enhance feature representation without incurring significant overhead. SFM integrates both channel and spatial attention into a unified, efficient structure. It combines a squeeze-and-excitation (SE) layer, which captures inter-channel dependencies, with a large context area (LCA) block that models spatial information using large kernel depthwise convolutions. The outputs from both branches are fused using element-wise multiplication, enabling the network to emphasize features based on both their spatial location and channel importance. By combining these complementary forms of attention, SFM provides a more robust and discriminative feature enhancement mechanism. This mechanism is particularly effective in visually degraded environments, such as nighttime underwater scenes, where object features are often subtle, and spatially scattered.

As shown in Figure 3, the SFM module comprises several components, beginning with a SE [29] block that assigns higher importance to the most informative feature maps by generating channel-wise probability weights. The output from the SE block is then normalized using layer normalization (LN), which stabilizes the feature distribution across batches and reduces the impact of redundant and outlier values. The normalized features are then forwarded into two parallel branches, both operating on the same input to extract complementary information for further refinement. To capture a wider receptive field while maintaining computational efficiency, one branch applies a depthwise convolution with a 7 × 7 kernel. On the other hand, the second branch applies a 1 × 1 convolution to model inter-channel relationships and introduce variation in feature transformation. This dual-branch structure enables SFM to effectively combine spatial and channel information, thereby enhancing feature representation. The outputs from both the LCA and 1 × 1 convolution branches are then combined using element-wise multiplication to merge spatial and channel-aware features. This fused representation is further refined through the feed forward with fusion (F3) module, which enhances feature mixing and reinforces channel relationships. The result is passed through a final 1 × 1 convolution to produce the output of the SFM module. The overall process can be defined as (5):

$$SFM(X) = Cv(F3(LCA(SEN(X))Conv_{1\times1}(SEN(X)))) \tag{5}$$

Squeeze and excitation module with normalization (SEN) is a widely adopted attention mechanism known for its ability to improve model performance by adaptively recalibrating channel-wise feature information. It utilizes average pooling to squeeze feature maps dimension into 1 × 1 × C and uses linear operation to

extract information based on channels. This attention module is applied at the beginning to assess the relevance of each feature map and assign higher weights to those containing more meaningful object information. This process can be illustrated as (6).

$$SE(X) = (W_2(ReLU(W_1(GAP(X))))) * X \tag{6}$$

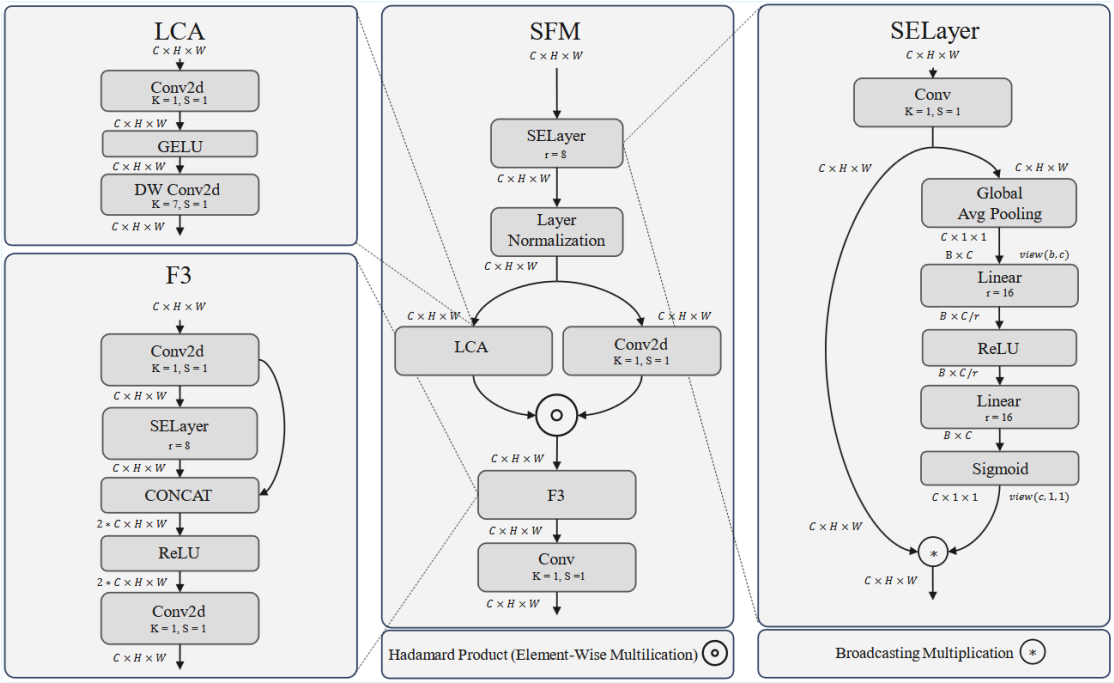


Figure 3. SFM, this module enhances feature representations by combining spatial and channel attention, allowing the network to focus on informative features under low-light conditions

Channel attention focuses on identifying which channels contain the most relevant information. GAP is an effective method for obtaining a representative value for each channel by computing the average of all spatial elements within a feature map, resulting in a single scalar value per channel. This operation captures global context and reflects the importance of each channel relative to the input. To further enhance feature representation, a two-layer fully connected network is applied after the pooling operation. The first linear layer reduces the dimensionality using a reduction ratio of 16 to minimize computational cost. A rectified linear unit (ReLU) activation is applied to suppress non-informative or negative values. The second linear layer restores the original channel dimension, followed by a sigmoid activation that generates attention weights. These weights guide the model to focus on channels that carry more informative features. This process enables the model to focus on the most informative features, making the subsequent feature extraction more efficient and effective in capturing critical object details. To preserve feature variation and ensure stable data distribution between each batches, the model incorporates LN immediately after the SE module. This operation can be described as (7).

$$SEN(X) = LN(SE(X)) \tag{7}$$

Similar to other channel attention mechanisms, the SE module reduces quality of spatial information, potentially discarding fine-grained details across extended spatial regions. To overcome this limitation, the design incorporates a LCA module, which is responsible for capturing and reinforcing spatial information through a dual-branch structure. The first branch applies a pointwise convolution to introduce feature variation and enrich the extracted representations. In parallel, the second branch uses a lightweight sequential convolution to capture a broader receptive field and strengthen spatial relationships. This process is formulated as (8):

$$LCA(X) = DW_{7\times7}(GeLU(Conv_{1\times1}(X))) \tag{8}$$

The LCA module aims to capture rich spatial context while minimizing computational cost. It begins with a 1 × 1 standard convolution ($Conv_{1\times1}$), followed by Gaussian error linear unit (GELU) activation. This step enhances inter-channel interactions and prepares the feature maps for more effective spatial processing. This non-linear activation is critical for refined feature selection. Unlike ReLU, which discards all negative values, GELU applies a smooth gating mechanism that allows small negative values to pass through. This mechanism helps receive small negative values that have potentially meaningful information. In order to enrich spatial understanding, a 7 × 7 depthwise convolution ($DW_{7\times7}$) is applied. The large kernel size allows the network to capture broader spatial dependencies, while the depthwise design helps control parameter growth and maintain computational efficiency. This combination enables the LCA module to model detailed local structures and long-range context efficiently. Furthermore, the outputs from the $Conv_{1\times1}$ and LCA branches are then fused using element-wise multiplication, reinforcing the most informative features and enhancing the overall feature representation. Additionally, to enhance feature interactions, the Feed Forward with Fusion ($F3$) module is employed. This module refines the fused representation by emphasizing informative features a nd reducing channel redundancy. It utilizes a single spatial convolution to strengthen inter-channel relationships, followed by an attention module that highlights channels containing the most relevant information based on the enhanced features. This sequence of operations is represented as (9):

$$F3(X) = Conv(ReLU(Concat[SE(Conv(X)), Conv(X)])) \tag{9}$$

The F3 module begins with a 1 × 1 convolution ($Conv$) to establish relationships between the input features, enhancing the representation of relevant information. It is followed by a SE block that recalibrates channel-wise importance and amplifies the channels that contain the most object-relevant information. The output of the SE block is concatenated with the ($Conv$) feature using a concatenation operation ([·]), effectively doubling the channel dimension and enabling richer feature interaction. The combined tensor serves as a unified representation for subsequent processing. To eliminate this irrelevant information, a ReLU function is applied, which removes all negative values. This operation retains only positive responses, allowing the network to focus on features that are more likely to contribute meaningful information. Finally, another 1 × 1 convolution ($Conv$) reduces the feature dimension back to its original size, producing a compact and enriched output that preserves both spatial and semantic information. This layered processing within the F3 module enables the network to adaptively fuse and refine features after attention, ensuring that the output of the SFM block is both discriminative and lightweight.

3.4. Neck

The feature map flow within a network is fundamental to effective learning and accurate prediction, particularly in object detection tasks. As illustrated in Figure 1, YOLO11-pico architecture maintains the core structure of its predecessor, YOLO11-nano, and adopts the path aggregation network (PAN) [30] as its neck component. The PAN structure plays a crucial role in fusing multi-scale feature maps generated by the backbone, allowing the model to integrate both low-level spatial details and high-level semantic context. The neck employs a concatenation approach to combine feature maps from different stages. This operation enables the preservation of detailed information from low-level layers, which is essential for detecting small objects. By facilitating a rich information flow between features of various scales, the PAN structure enhances the diversity of the feature representations passed to the detection head. This mechanism enables better localization and classification performance while maintaining a compact and efficient architecture.

3.5. Head

YOLO11-pico retains the detection head structure and loss functions from the original YOLO11-nano. The head features two parallel branches, separating bounding box regression and object classification tasks. This decoupled design allows the model to optimize each task independently, while maintaining efficiency and suitability for real-time and edge deployment. The detection head combines standard and depthwise convolutional layers. In the bounding box regression branch, it applies two 3×3 standard convolutions, each followed by batch normalization and a SiLU activation. A final 1×1 convolutional layer generates the bounding box predictions. It produces tensor outputs at three spatial resolutions, where each scale corresponds to a different level of feature representation, from low-level to high-level features. Each output tensor has a channel dimension equal to four times the value of regmax. The bounding box regression predicts four values representing the diagonal coordinates of the bounding box, specifically xmin, ymin, xmax, and ymax. The regmax parameter defines the maximum range for regression and is set to 16 by default.

The classification branch is responsible for predicting object classes and uses a deeper convolutional structure compared to the regression branch. It applies two sequential operations, each consisting of a 3×3 depthwise convolution followed by batch normalization and SiLU activation, and then a 1×1 standard convolution that is also followed by batch normalization and SiLU activation. Additionally, a final 1×1 convolution layer is placed at the end to produce the classification output. Similar to the bounding box branch, it generates outputs at three grid sizes. However, the number of output channels in the classification branch is set equal to the number of classes to represent class probabilities efficiently. This configuration captures spatial and channel-level features effectively while keeping the computational load low. The training stage utilizes a composite loss function that combines complete intersection over union (CIoU) for precise localization [31], distribution focal loss (DFL) [32] for distributed bounding box prediction, and binary cross-entropy (BCE) [33] for classification supervision.

3.6. Dataset

This study utilizes a dataset from previous work [3], collected in Bunaken Marine Park for the development of a sea turtle detection model. The dataset includes annotated images of three sea turtle species, namely green, hawksbill, and olive ridley, captured under various underwater conditions. The dataset is augmented using a handcrafted process with image editing software to simulate nighttime scenes by reducing brightness to -124, setting contrast to zero, adjusting exposure to -4.19, and applying a gamma of one. It also incorporates localized lighting effects that mimic artificial illumination, as shown in Figure 4. These augmentations help replicate the low-light conditions commonly encountered during nighttime monitoring. The dataset used in this study includes real (non-augmented) nighttime images captured directly from Bunaken waters under natural low-light conditions. These images reflect real-world challenges such as limited illumination, underwater noise, and visibility degradation. Although augmentation techniques were applied to enhance robustness, the model is primarily trained on authentic nighttime data to ensure realistic feature representation.

The baseline dataset was collected from multiple dive sites within Bunaken National Park. These sites were selected based on the regional sea turtle distribution map, which identifies areas with high turtle activity and biodiversity. These locations represent ecologically diverse underwater environments, varying in reef structures, water clarity, and natural lighting conditions. Subsequently, the dataset is annotated manually using a bounding-box tool, generating YOLO formatted labels that include the class label and normalized coordinates (x, y, w, h). The complete dataset comprises 48,244 annotated images, totaling 5.4 GB, with varying resolutions of 3840 × 2160, 1920 × 1080, and 1080 × 720. The dataset is divided into three subsets to support model development. Out of the total images, 30,934 were allocated for training, 7,698 for testing, and 9,612 for validation. The dataset comprises 28,053 images of *Chelonia mydas* (green turtle), 11,382 of *Eretmochelys imbricata* (hawksbill turtle), and 8,809 of *Lepidochelys olivacea* (olive ridley turtle). It is locally collected and class-imbalanced, and no data augmentation is applied to increase the pattern recognition challenge. The training set enables the model to learn and update its parameters, while the validation set guides hyperparameter tuning and performance tracking during training. The test set provides a final benchmark to evaluate how well the model generalizes to unseen data. This structured split supports a balanced and reliable learning process.

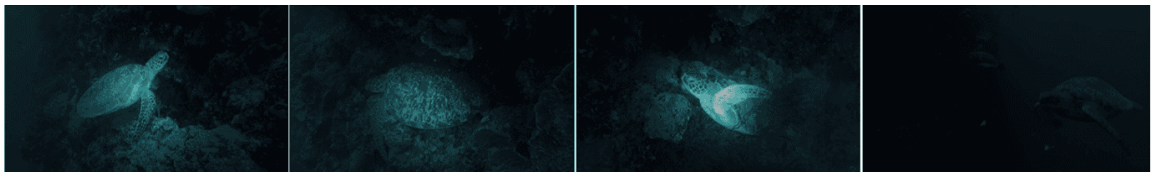


Figure 4. Visualization of underwater dataset augmentation. Handcrafted augmentations simulate nighttime conditions by reducing brightness, adjusting contrast, and adding lighting effects

3.7. Training configuration

In order to ensure a fair comparison, the training process does not employ any transfer learning methods. The NiteTron model is trained for 150 epochs, which is identified as the optimal configuration based on empirical analysis, as training beyond this point yields diminishing performance gains and increases the risk of vanishing gradients. Furthermore, both training and evaluation used an input resolution of 640 × 640

pixels, following the default YOLO settings. The stochastic gradient descent (SGD) is employed to optimize the learning process on large datasets, combined with a learning rate of 0.01 and a batch size of 16 to promote stable convergence. These configurations provided a practical balance between accuracy and computational efficiency, enabling effective training on moderately powered hardware. The training process utilizes an NVIDIA GeForce RTX 3060 graphics processing unit (GPU) with 12 GB of video random-access memory (VRAM), supported by an AMD Ryzen 5 3600 6-core CPU and 16 GB of RAM. The system runs on Ubuntu 22.04 and is configured with Compute Unified Device Architecture (CUDA) version 12.4 to support GPU acceleration. The model architecture leverages the PyTorch deep learning framework. This setup provides sufficient computational resources to process high-resolution image inputs, enabling faster training iterations.

3.8. Inference configuration

In order to simulate deployment on low-edge devices, this study evaluates the NiteTron on two different hardware configurations to assess real-time performance and deployment feasibility. The first device, a compact mini-PC built for low-power applications and embedded use cases, features an Intel N5105 processor running at a base clock speed of 2.0 GHz. It includes 8 GB of low-power double data rate 4 (LPDDR4) RAM and 256 GB of internal storage. This configuration represents a resource-constrained environment, making it suitable for evaluating the performance efficiency, responsiveness, and practicality in edge deployments where power, memory, and processing capabilities are limited. The second device is an all-in-one desktop system equipped with an Intel Core i7-12700 processor with 20 cores, 16 GB of RAM, and the Ubuntu Linux operating system. This setup provides a more powerful and balanced computing environment, enabling high-throughput inference and reduced latency.

4. RESULT AND DISCUSSION

4.1. Ablation study

This section presents an ablation study to systematically investigate the individual contribution of each proposed module to overall model performance. As shown in the Table 1. This study evaluates the impact of each proposed module by comparing several modified YOLO11-based model variants. The original YOLO11-nano serves as the baseline model, achieving a test mAP 95 of 0.653 with 2.59 million parameters and 6.4 GFLOPs. In comparison, YOLO11-pico with channel reduction decreases the parameter count to 1.49 million and the computational cost to 5.5 GFLOPs. This strategy reduces the number of parameters by 42.13% relative to the baseline and lowers the computational cost by 14.06. Moreover, YOLO11-pico achieves an mAP 95 of 0.660, representing a 1.07 improvement over the baseline score of 0.653. It demonstrates that the proposed backbone is more efficient than the original while maintaining competitive detection precision. YOLO11-pico is an optimized variant of YOLO11-nano, designed by adjusting the number of channels in several convolutional blocks within the YOLO11 architecture. The structure aims to reduce redundancy in the feature maps, where overlapping or repeated information can make it more difficult for the model to distinguish between relevant object features and background noise. By minimizing this redundancy, the model not only enhances its ability to extract discriminative features but also reduces computational complexity.

Table 1. Ablation study results comparing efficiency and detection performance mAP on validation and test sets

Model	Parameters	GFLOPs	Val		Test	
			mAP 50	mAP 95	mAP 50	mAP 95
YOLO11n	2,590,425	6.4	0.845	0.723	0.782	0.653
YOLO11p	1,498,905	5.5	0.829	0.712	0.786	0.660
YOLO11p+	1,435,481	5.5	0.826	0.703	0.795	0.664
YOLO11p+_S2C	1,450,361	5.5	0.827	0.700	0.809	0.667
YOLO11p+_S2C_EC	1,450,361	5.5	0.829	0.709	0.798	0.670
YOLO11p+_S2C_SFM w/o F3	1,661,033	5.7	0.802	0.693	0.788	0.676
Proposed	1,957,241	5.9	0.832	0.710	0.802	0.682

Another experiment involving the YOLO11p+ variant demonstrates improved performance after removing the two convolutional with partial self-attention (C2PSA) modules. This modification achieves a mAP of 0.664, utilizing only 1.43 million parameters, indicating a more efficient model configuration. The ablation result shows that the inclusion of the C2PSA module, which introduces additional complexity through

self-attention mechanisms [34], is less effective in extracting high-level semantic features specific to sea turtle detection. The performance continues to improve with the addition of each component into the model. Incorporating the S2C module with depthwise convolutions [35] increases the accuracy to 0.667 and demonstrates its effectiveness in extracting features. Furthermore, adding the EC module adapted from global sigmoid linear unit (GSILU) [36] provides similar performance while improving computational efficiency. However, removing the feed forward with fusion (F3) from the SFM structure decreases the performance to 0.676. It shows that the F3 module plays a crucial role in EC fusion and improving overall localization performance. The complete model integrates both S2C and SFM modules, which achieves the highest mAP 95 of 0.682 on the test subset dataset. It generates 1.96 million parameters. Despite having fewer parameters than YOLO11-nano, it delivers better performance across all validation and test metrics. This result confirms that the combined use of S2C and SFM effectively enhances both spatial and channel-wise representations. The final model strikes a strong balance between accuracy and efficiency, making it well-suited for deployment in resource-constrained and real-time applications, such as nighttime underwater turtle detection.

In order to evaluate model consistency, Table 2 presents results from three independent training runs, reporting the mean (μ) and standard deviation ($\pm$) for each performance. Furthermore, these runs assess the statistical significance of improvements using the p-value from mAP 95 with a threshold of 0.05. A p-value higher than the threshold indicates that the improvement is not statistically significant. The YOLO11n as the baseline achieves mAP 50 (μ) of 0.657 with a standard deviation of 0.003. Furthermore, the YOLO11p+ achieves an mAP 50 of 0.667 with a standard deviation of 0.002, and its p-value of 0.004 shows a statistically significant improvement. YOLO11p with complete S2C achieves 0.669 with 0.001 standard deviation, and the p-value of 0.251 indicates no statistically significant improvement. The proposed model with SFM achieves the highest mAP 95 of 0.682 with zero standard deviation. The p-value of 0.0007 indicates a statistically significant improvement.

Table 2. Consistency analysis averaged over three independent training runs

Model	Parameters	Val		Test		p-value
		mAP 50 (μ)	mAP 95 (μ)	mAP 50 (μ)	mAP 95 (μ)	(mAP 95)
YOLO11n	2,590,425	0.816 ($\pm$0.025)	0.701 ($\pm$0.019)	0.789 ($\pm$0.006)	0.657 ($\pm$0.003)	
YOLO11p+	1,435,481	0.838 ($\pm$0.01)	0.715 ($\pm$0.01)	0.801 ($\pm$0.005)	0.667 ($\pm$0.002)	0.004
YOLO11p+ with S2C	1,450,361	0.832 ($\pm$0.003)	0.709 ($\pm$0.0)	0.800 ($\pm$0.002)	0.669 ($\pm$0.001)	0.251
Proposed	1,957,241	0.832 ($\pm$0)	0.710 ($\pm$0)	0.802 ($\pm$0)	0.682 ($\pm$0)	0.0007

4.2. Evaluation on dataset

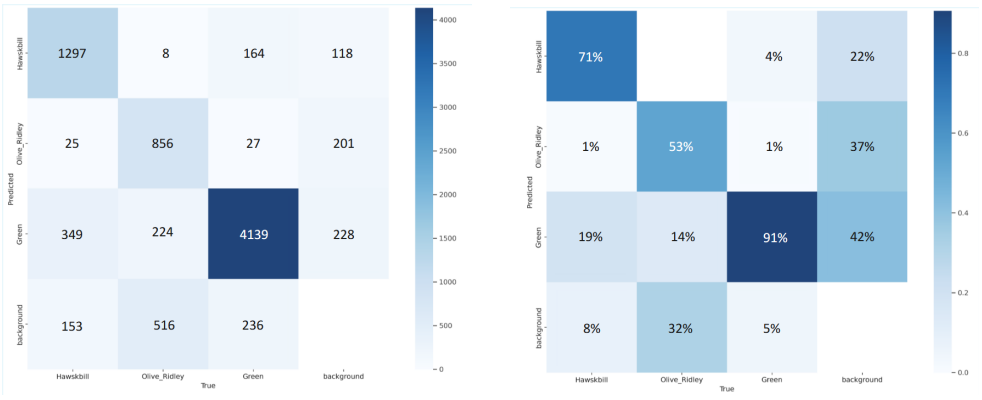
In order to comprehensively evaluate the effectiveness and efficiency of the NiteTron, this study applies consistent training and evaluation settings across all experiments. As shown in Table 3, the comparison includes faster R-CNN, EfficientDet, YOLO11n combined with swin transformer, YOLO11n combined with ResNet50, multiple YOLO variants, and DETR. It provides a consistent benchmark for comparing models with different complexities. The faster R-CNN [37] model achieves an mAP 95 of 0.26 with 14.7 M parameters and 50.4 GFLOPs, showing the lowest performance among the compared networks. Subsequently, the EfficientDet [38] achieves an mAP 95 of 0.585 with 3.8 million parameters and 2.2 GFLOPs, making it the weakest performer despite its lightweight design. Furthermore, DETR [39] achieves an mAP 95 of 0.565 but uses over 32 M parameters and 108 GFLOPs. YOLO11n with a swin transformer [40] backbone achieves an accuracy of 0.641 mAP 95 and has a computational cost of 47.9 GFLOPs. In contrast, YOLO11n with a ResNet50 [41] backbone achieves mAP 95 of 0.663 and generates 6.2 GFLOPs. YOLOv3-tiny also requires substantial computational resources, with 12 million parameters and 19.0 GFLOPs, but it only achieves a test mAP 95 of 0.597. More recent YOLO versions, including YOLOv5n, YOLOv6n [42], and YOLOv8n, offer improved accuracy with fewer resources. In addition, YOLOv5n achieves a mAP 95 test of 0.643 with 2.5 million parameters and 7.2 GFLOPs. YOLOv6n reaches 0.684 mAP 95 with 4.2 million parameters, while YOLOv8n shows similar accuracy but with fewer GFLOPs. In the comparison, YOLOv9-tiny achieves a test mAP 95 of 0.648 with 7.9 GFLOPs, striking a balance between accuracy and computational demand, but still falling short of proposed model performance.

YOLOv10-nano [10] achieves a mAP 95 of 0.649 with 8.4 GFLOPs, reflecting slightly higher performance than YOLOv9. However, it remains less efficient than YOLO11-pico in metrics of both accuracy and computational cost. The lightweight YOLO11 offers further improvements. YOLO11n as a baseline, achieves 0.653 mAP 95 with 2.59 million parameters. YOLO11p and YOLO11p+ (without C2PSA) reduce the number

of parameters while improving accuracy to 0.660 and 0.664, respectively. These results highlight the benefits of using a more compact backbone. The proposed network integrates an efficient channel and enhancement module, achieving the highest test mAP 95 of 0.682 with only 1.96 million parameters and 5.9 GFLOPs. Compared to the YOLO11n baseline, it reduces the parameter count by over 600 M while improving mAP 50 by 2.56% and mAP 95 by 4.44%, demonstrating a superior balance between accuracy and computational efficiency. The results confirm that the lightweight design and added modules are effective. The proposed model also surpasses the previous work [3] on the same dataset. In addition, the modified YOLO11 offers strong accuracy while remaining efficient, making it reliable for real-time detection on devices with limited computing power. The confusion matrix in Figure 5 shows NiteTron's performance in classifying hawksbill, olive ridley, green, and background. In Figure 5(a) presents the non-normalized confusion matrix, while Figure 5(b) shows the normalized version. High diagonal values indicate correct classifications, while off-diagonal values show misclassifications.

Table 3. Comparison of model parameters, GFLOPs, and detection performance (mAP) on validation and test sets

Model	Parameters	GFLOPs	Validation		Test	
			mAP 50	mAP 95	mAP 50	mAP 95
Faster R-CNN	14,714,688	50.4	0.529	0.325	0.450	0.260
EfficientDet	3,787,355	2.2	0.751	0.614	0.755	0.585
DETR	32,812,241	108	0.730	0.627	0.651	0.565
YOLOv11n+swin transformer	2,734,755	47.9	0.806	0.686	0.762	0.641
YOLOv11n+ResNet50	2,176,761	6.2	0.820	0.698	0.787	0.663
YOLOv3-tiny	12,133,670	19.0	0.780	0.624	0.744	0.597
YOLOv5n	2,509,049	7.2	0.829	0.707	0.771	0.643
YOLOv6n	4,238,441	11.9	0.837	0.722	0.818	0.684
YOLOv8n	3,011,433	8.2	0.838	0.722	0.789	0.669
YOLOv9-tiny	2,005,993	7.9	0.820	0.708	0.711	0.648
YOLOv10n	2,708,210	8.4	0.812	0.699	0.769	0.649
YOLOv12n	2,568,633	6.5	0.798	0.687	0.770	0.647
YOLO11n	2,590,425	6.4	0.845	0.723	0.782	0.653
YOLO11p	1,498,905	5.5	0.829	0.712	0.786	0.660
YOLO11p+	1,435,481	5.5	0.826	0.703	0.795	0.664
Proposed	1,957,241	5.9	0.832	0.710	0.802	0.682



(a) (b)

Figure 5. Confusion matrices of the proposed model: (a) non-normalized and (b) normalized, showing classification performance across three categories with strong diagonal values indicating correct predictions

The green turtle class records the highest number of true positives, with 4,139 samples correctly classified and a 91% true positive rate. However, the model incorrectly predicts 349 hawksbill samples as green Turtle. It also misclassifies 224 olive ridley samples and 228 background samples as green turtle. These errors indicate visual similarity among the categories. For the hawksbill class, the model correctly identifies

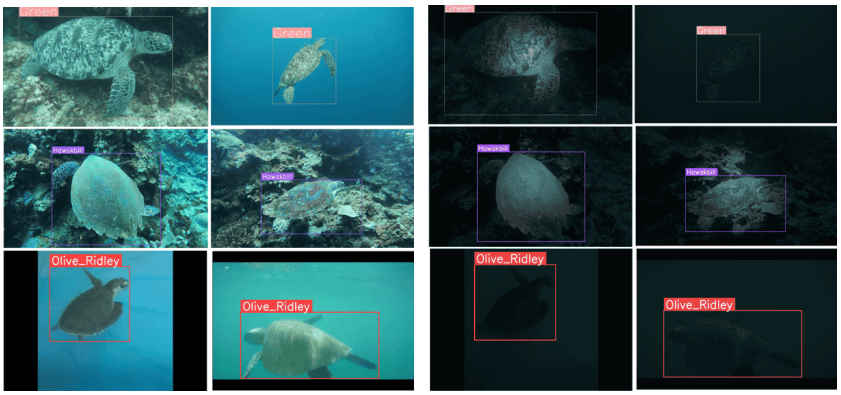
1,297 samples. However, it misclassifies 4% of green samples and 22% of background samples as hawksbill, which suggests partial feature overlap. The olive ridley class achieves a 53% true positive rate with 856 correct predictions. Misclassifications in this class mainly originate from background, green, and hawksbill samples. In addition, the background class receives false predictions from all turtle classes. Specifically, 32% of olive ridley samples, 5% of green samples, and 8% of hawksbill samples are incorrectly labeled as background.

NiteTron exhibits the highest error rate in the Green class, with a misclassification rate of 42%, corresponding to 228 incorrect predictions. This performance degradation is primarily attributed to nighttime conditions, where limited illumination reduces the availability of discriminative texture and shape information. In contrast, the hawksbill class exhibits the lowest error, with only eight misclassified samples. This species possesses distinctive morphological features that provide strong visual cues for accurate recognition. Several remaining misclassifications are caused by challenging underwater conditions such as low contrast, partial occlusion, and color blending with the surrounding environment [43], [44]. These factors degrade image quality and limit the model's ability to clearly distinguish turtles from the background.

As shown in Table 4, the proposed network outperforms YOLO11n on the daytime dataset, achieving an mAP 95 of 0.696. This result corresponds to a 2.5% improvement over YOLO11n, which attains an mAP 95 of 0.679. In addition to mAP, the performance is further evaluated using standard object detection metrics, including precision, recall, and F1-score. Figure 6 further illustrates detection results under both daytime and nighttime augmented conditions. As shown in Figure 6(a), the proposed model achieves stable and consistent detections in daytime scenarios. In contrast, Figure 6(b) shows that nighttime conditions introduce reduced visibility. Nevertheless, the model continues to reliably detect sea turtles, demonstrating strong robustness under low-illumination environments. Furthermore, this study evaluates the robustness of the proposed detector under varying illumination conditions in the test data, including 5,879 instances under dark conditions and 3,733 images with spotlight illumination scenarios. As shown in Table 5, the proposed detector achieves mAP improvements of 18.10% and 18.32% over the original network under dark and spotlight conditions, respectively. In addition, it consistently outperforms the baseline in terms of precision, recall, and F1-score, demonstrating enhanced robustness to illumination variations. Figure 7 illustrates detection results in real nighttime underwater scenarios without any image processing or augmentation, where low illumination and backscatter increase recognition difficulty. Despite these challenging nighttime conditions, the proposed model successfully detects sea turtles. These results demonstrate that data augmentation enhances generalization to real nighttime underwater environments.

Table 4. Comparison of detection performance under daytime and nighttime conditions

Model	Daytime				Nightime			
	Precision	Recall	F1-score	mAP 95	Precision	Recall	F1-score	mAP 95
YOLO11-nano	0.839	0.695	0.760	0.679	0.814	0.704	0.755	0.653
Proposed	0.844	0.696	0.763	0.696	0.845	0.700	0.766	0.682



(a) (b)

Figure 6. Visualization of detections on (a) daytime images and (b) nighttime-augmented images created through brightness reduction and contrast adjustment

Table 5. Comparison of detection performance under different lighting conditions

Model	Dark				Spotlight			
	Precision	Recall	F1-score	mAP 95	Precision	Recall	F1-score	mAP 95
YOLO11-nano	0.758	0.564	0.647	0.536	0.841	0.709	0.769	0.633
Proposed	0.782	0.653	0.712	0.633	0.942	0.800	0.865	0.749

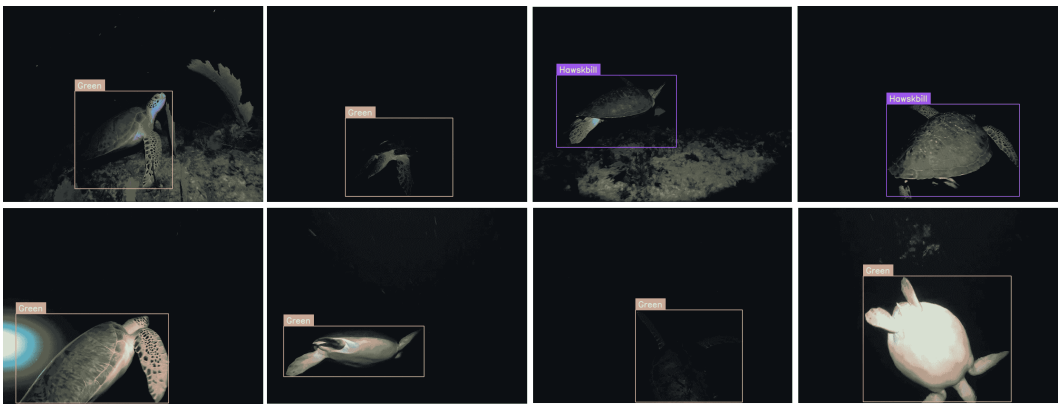


Figure 7. Visualization of detection results under real nighttime underwater conditions, illustrating the effects of low illumination and natural lighting variations in actual scenes

4.3. Heatmap evaluation

The heatmaps shown in the Figure 8 provide a visual explanation of how the proposed model identifies and localizes different sea turtle species. These visualizations are using Eigen class activation map (EigenCAM) [45], a gradient-free class activation mapping method that highlights the most influential regions in the image contributing to the model's decision. The analysis of activation maps reveals that the model focuses on the entire shell and flipper regions for the green turtle class, indicating attention to key structural and textural features distinctive to the species. The model demonstrates accurate detection and classification, with activations concentrated around the body despite variations in posture and lighting conditions.

On the other hand, the activation maps for hawksbill examples highlight the central region of the shell, where the distinctive overlapping scutes and sharper edges characteristic of the species are most evident. The heatmaps indicate that the model effectively recognizes these specific features, even in visually complex environments where the turtle partially blends into the background. In the third case, the model's attention is concentrated on the silhouette and flippers of the olive ridley turtle. Despite low-contrast underwater conditions, the heatmaps reveal that the model effectively captures outline and subtle shape cues that differentiate this species from others. This demonstrates the model's robustness under challenging visual conditions and its ability to focus on biologically meaningful features.

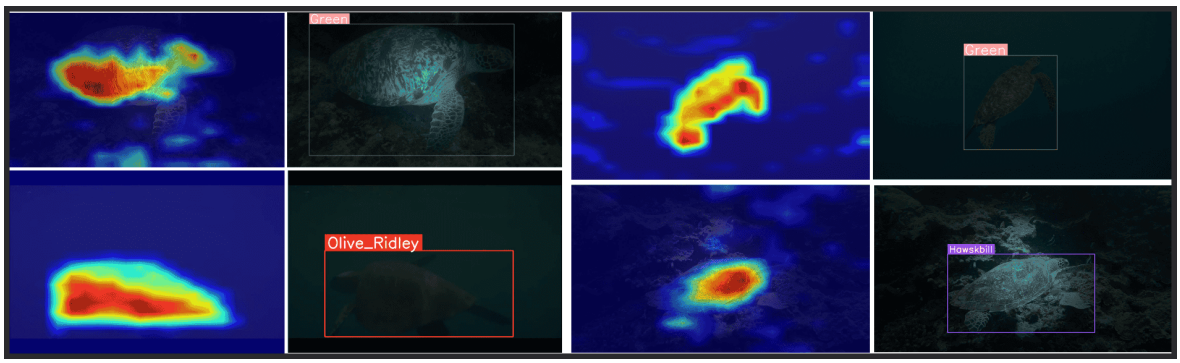


Figure 8. Heatmap visualization of the proposed model. Highlights the regions most influential to the model's predictions for each sea turtle class using EigenCAM

4.4. Runtime efficiency

In order to evaluate runtime efficiency, the proposed model runs on two CPU platforms to assess its real-time performance. An Intel Core i7-12700 and a low-power Intel N5105 represent personal computer (PC) desktop and mini PC devices. As shown in Figure 9, the modified channel YOLO11-pico achieves 23.39 FPS on the PC desktop and 2.95 FPS on the mini PC. Despite having only 1.5 million parameters and 5.5 GFLOPs, the model offers an optimal trade-off between speed and accuracy. DETR shows the slowest speed due to its large size of 32.8 million parameters and 108 GFLOPs, reaching only 1.67 FPS on both platforms. Although YOLOv3-tiny is smaller than many other models, it still requires 19 GFLOPs and gives relatively low performance, with a mAP of just 0.597 and an inference speed of 8.75 FPS. Recent YOLO variants demonstrate better efficiency. YOLOv5n runs at 21.64 FPS and 2.50 FPS with 0.643 mAP. YOLOv6n and YOLOv8n offer improved accuracy of 0.684 and 0.669, but with higher computational demands and slightly lower speeds. YOLOv9-tiny and YOLOv10n maintain balanced performance in both speed and accuracy, while YOLOv12n [46] proves efficiency without sacrificing accuracy. Channel tuning directs the model to focus on the most relevant features, improving accuracy while reducing computation. The SFM module further refines channel selection, allowing the model to prioritize important information more effectively.

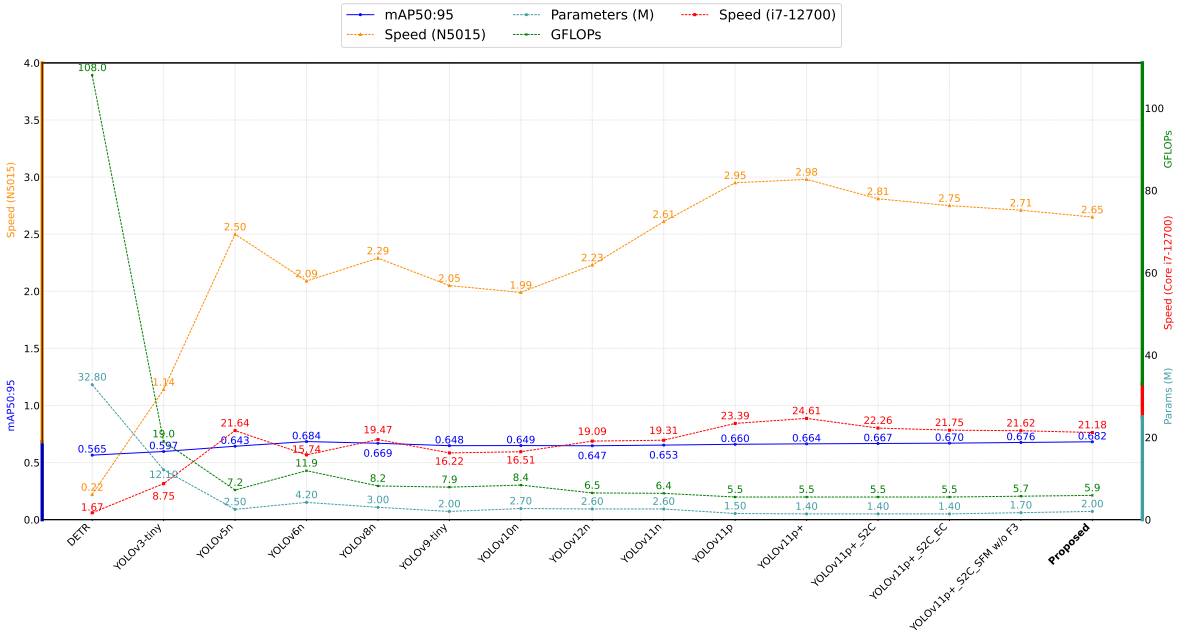


Figure 9. Runtime efficiency of the models. Shows the relative speed and computational performance of each model during inference

4.5. Discussion

NiteTron focuses on localizing and classifying Bunaken's unique turtle categories by integrating S2C and SFM. This design enables the model to accurately detect and classify turtles while outperforming other lightweight detectors. The combined modules enhance feature quality and computational efficiency, addressing the limitations of the original YOLO11n architecture, which does not explicitly enhance global feature relationships. By incorporating these mechanisms, NiteTron delivers superior detection capability and more reliable feature extraction, achieving higher precision without increasing computational complexity. SFM strengthens the model's ability to capture feature relationships under limited illumination by employing an enhancement module that adaptively modulates feature importance. Functionally, it operates similarly to an efficient transformer, enabling the lightweight network to highlight salient features without introducing substantial additional parameters or computational overhead. In comparison, the Swin Transformer exhibits lower performance when inserted into the backbone. Although it incorporates long-range dependencies and positional encoding, its window-based mechanism restricts feature extraction to local regions, thereby limiting its ability

to capture fully global interactions. This approach also introduces a high computational cost. In contrast, the proposed SFM module performs a sequential integration of global and spatial feature interactions, enabling more comprehensive representation learning with significantly lower complexity. SFM identifies feature correspondences using Hadamard product at the same spatial coordinates across two feature maps with different frequency characteristics. It employs channel attention in the initial block to amplify these responses, enabling more accurate modeling of global contextual information while maintaining computational efficiency. Furthermore, S2C emphasizes lightweight feature extraction without relying on deep or complex convolutional layers. Since YOLO11n does not incorporate efficient convolutional operations, such as depthwise convolutions. The baseline model still maintains a relatively high parameter cost and computational complexity. In contrast, the proposed S2C block leverages the enhancement channel to significantly reduce computational cost and the number of trainable parameters, while preserving detection precision and preventing noticeable performance degradation.

Although the proposed sea turtle detection model demonstrates strong performance on the Bunaken Marine Park dataset, its transferability to other underwater environments may be limited. Variations in water turbidity, color attenuation, illumination conditions, seafloor composition, camera hardware, and marine biodiversity can introduce domain shifts that significantly challenge detection performance. The visual characteristics of sea turtles generally remain consistent across different regions, enabling reliable species differentiation [20]. Green turtles typically exhibit greenish coloration, hawksbill have scaly, broad, brown-toned shells, and olive ridley are characterized by their large, rounded shape and gray coloration. Although coastal environments may introduce slight variations in appearance, these primary morphological cues remain sufficiently distinctive for the deep learning model to reliably recognize each species. Another challenge arises when augmented training data must be evaluated under real-world conditions [22]. However, our experiments demonstrate that this limitation does not hinder NiteTron's ability to accurately localize turtle species, as illustrated in Figure 7. Despite being trained on data from the Bunaken Marine Park, the proposed model achieves reliable detection in other underwater environments, demonstrating its transferability across varying conditions.

On the other hand, this work also has class limitations that focus specifically on Bunaken's endemic species, including green, hawksbill, and olive ridley turtles. Nevertheless, the model has strong potential for successful application to additional classes or other turtle species. This is supported by the fact that most turtle species exhibit clear and distinguishable visual characteristics, which can enable reliable feature learning and accurate classification when the model is adapted or retrained on new categories. Future work will involve evaluating the model across multiple marine regions, incorporating domain adaptation strategies, and expanding the training dataset with more diverse underwater imagery and additional turtle classes to further enhance robustness and generalizability.

5. CONCLUSION

This study presents NiteTron, a lightweight real-time detector for nighttime sea turtle recognition in underwater environments. The optimized YOLO11-pico architecture incorporates two key modules, including the SFM and the S2C. These additional modules enhance the model's ability to extract meaningful features in low-light and visually challenging conditions, while maintaining a compact and efficient design. Experimental results demonstrate that the proposed model achieves a mAP @0.95 of 0.682 while maintaining low computational complexity, outperforming other lightweight YOLO variants. With real-time speeds of 21.18 FPS on a desktop CPU and 2.65 FPS on a low-power mini PC, the model proves suitable for practical deployment in resource-constrained environments. These findings highlight its potential for continuous, low-impact monitoring in underwater conservation efforts. Compared to the YOLO11n baseline, NiteTron generates 1.9 million parameters, which is a 24.45% reduction from the original. It also shows improved computational efficiency, requiring only 5.9 GFLOPs, which is 7.81% less than the baseline. Despite, the model achieves higher performance, with mAP 50 and mAP 50:95 scores of 0.802 and 0.682, respectively. These performance represent improvements of 2.56% and 4.44% over the baseline, confirming the effectiveness of the proposed model. The proposed model not only enhances detection accuracy but also significantly improves computational efficiency, making it well-suited for real-time deployment in both desktop and resource-constrained environments. Future work will focus on loss function development to further improve performance without sacrificing inference speed. In addition, improving cross-environment generalization will be prioritized through multi-location dataset expansion and domain adaptation strategies.

FUNDING INFORMATION

This research is funded by Applied Research of UNSRAT Excellence (RTUU) Sam Ratulangi University 2025 with contract number 638/UN12.27/LT/2025.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Muhamad Dwisnanto Putro	✓	✓		✓	✓		✓		✓	✓				
Jane Litouw		✓		✓		✓			✓		✓	✓		
Abdul Haris Ontowirjo	✓	✓			✓	✓			✓			✓		
Sherwin Reinaldo U. Sompie			✓	✓		✓	✓	✓		✓				
Hebron Prasetya	✓		✓			✓		✓		✓	✓		✓	

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware Programing
Va : **Va**lidation
Fo : **Fo**rmal Analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject Administration
Fu : **Fu**nding Acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are openly available in https://bit.ly/4hSScMU.

REFERENCES

[1] N. Noguchi *et al.*, "Efficient wildlife monitoring: Deep learning-based detection and counting of green turtles in coastal areas," *Ecological Informatics*, vol. 86, p. 103009, 2025, doi: 10.1016/j.ecoinf.2025.103009.
[2] M. D. Putro, Y. Mose, A. C. Andaria, J. Litouw, V. C. Poekoel, and X. Najoan, "Streamlining Deep Learning Network for Real-Time Sea Turtle Detection," *Jurnal Rekayasa Elektrika*, vol. 20, no. 3, pp. 116–124, 2024, doi: 10.17529/jre.v20i3.35236.
[3] M. D. Putro, D. G. S. Ruindungan, R. Syahputra, T.-H. Oh, I. Y. Chun, and V. C. Poekoel, "An Efficient and Effective Sea Turtle Detection Using Positioning Enhancement Module," in *Proc. 2024 Int. Workshop on Intelligent Systems (IWIS)*, 2024, pp. 1–6, doi: 10.1109/IWIS62722.2024.10706029.
[4] H. Xu, W. Zhao, and G. Liu, "Research on Turtle Target Detection System Based on YOLO Algorithm," *2024 6th International Conference on Internet of Things, Automation and Artificial Intelligence (IoTAAI)*, pp. 376–380, Jul. 26, 2024. doi: 10.1109/iotaai62601.2024.10692385.
[5] M. D. Putro, A. Sutrisno, I. S. Manembu, I. Y. Chun, and T.-H. Oh, "STAR: Sea Turtle Basic Activity Recognizer Network via Efficient Transformer," *IEEE Access*, vol. 13, pp. 171356–171370, 2025, doi: 10.1109/access.2025.3615067.
[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.
[7] R. Khanam and M. Hussain, "YOLOv11: An Overview of the Key Architectural Enhancements," *arXiv*, 2024, doi: 10.48550/ARXIV.2410.17725.
[8] D. Liao, Y. Shen, J. Ou, and Y. Du, "Research on Underwater Target Detection Technology Based on SMV-YOLOv11n," *IEEE Access*, vol. 13, pp. 119820–119830, 2025, doi: 10.1109/ACCESS.2025.3584313.
[9] J. Ou and Y. Shen, "Underwater Target Detection Based on Improved YOLOv7 Algorithm With BiFusion Neck Structure and MPDIoU Loss Function," *IEEE Access*, vol. 12, pp. 105165–105177, 2024, doi: 10.1109/ACCESS.2024.3436073.
[10] S. Sriram, A. P., A. K. T. P., N. Vijayaraj and T. Murugan, "Enhanced YOLOv10 Framework Featuring DPAM and DALSM for Real-Time Underwater Object Detection," *IEEE Access*, vol. 13, pp. 8691–8708, 2025, doi: 10.1109/ACCESS.2025.3527315.
[11] Q. Tang, Z. Wang, X. Wang, and S.-W. Zhang, "AFHNet: Attention-Free Hybrid Network for Salient Object Detection in Underwater Images," *IEEE Access*, vol. 13, pp. 89287–89299, 2025, doi: 10.1109/ACCESS.2025.3567606.
[12] J. Huang, C. Fang, X. Zheng, and J. Liu, "YOLOv8-UC: An Improved YOLOv8-Based Underwater Object Detection Algorithm," *IEEE Access*, vol. 12, pp. 172186–172195, 2024, doi: 10.1109/ACCESS.2024.3496925.
[13] S. Manimurugan *et al.*, "HLASwin-T-ACoat-Net Based Underwater Object Detection," *IEEE Access*, vol. 12, pp. 32200–32217, 2024, doi: 10.1109/ACCESS.2024.3369063.

[14] T.-N. Pham, V.-H. Nguyen, K.-R. Kwon, J.-H. Kim, and J.-H. Huh, "Improved YOLOv5 Based Deep Learning System for Jellyfish Detection," *IEEE Access*, vol. 12, pp. 87838–87849, 2024, doi: 10.1109/ACCESS.2024.3405452.

[15] X. Li, Y. Wang, Y. Zhao, and G. Chen, "UW-DETR: Feature Fusion Enhanced RT-DETR for Improving Underwater Object Detection," *IEEE Access*, vol. 12, pp. 191967–191979, 2024, doi: 10.1109/ACCESS.2024.3515960.

[16] Z. Li, H. Xie, J. Feng, Z. Wang, and Z. Yuan, "YOLOv7-PE: A Precise and Efficient Enhancement of YOLOv7 for Underwater Target Detection," *IEEE Access*, vol. 12, pp. 133937–133951, 2024, doi: 10.1109/ACCESS.2024.3417322.

[17] M. Zhang, Z. Wang, W. Song, D. Zhao, and H. Zhao, "Efficient Small-Object Detection in Underwater Images Using the Enhanced YOLOv8 Network," *Applied Sciences*, vol. 14, no. 3, art. no. 1095, 2024, doi: 10.3390/app14031095.

[18] Y. Jin *et al.,* "A Novel Method for the Object Detection and Weight Prediction of Chinese Softshell Turtles Based on Computer Vision and Deep Learning," *Animals*, vol. 14, no. 9, 2024, doi: 10.3390/ani14091368.

[19] V. Y. Chen, Y.-W. Wu, C.-W. Hu, and Y.-S. Han, "Enhancing green sea turtle (Chelonia mydas) conservation for tourists at Little Liuqiu island, Taiwan: Application of deep learning algorithms," *Ocean & Coastal Management*, vol. 252, p. 107111, Jun. 2024, doi: 10.1016/j.ocecoaman.2024.107111.

[20] J.-W. Baek, J.-I. Kim, and C.-B. Kim, "Deep learning-based image classification of sea turtles using object detection and instance segmentation models," *PLoS ONE*, vol. 19, no. 11, pp. 1–15, 2024, doi: 10.1371/journal.pone.0313323.

[21] M. Liu, W. Zhang, C. Wei, Z. Bao, J. Hu, and J. Zhou, "PLGAT: Underwater Plectropomus leopardus Recognition Using Global Attention Mechanism and Transfer Learning," *IEEE Access*, vol. 12, pp. 185149–185159, 2024, doi: 10.1109/ACCESS.2024.3509622.

[22] L. Adam, V. Čermák, K. Papafitsoros and L. Picek, "SeaTurtleID2022: A long-span dataset for reliable sea turtle re-identification," *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024, pp. 7131-7141, doi: 10.1109/WACV57701.2024.00699.

[23] H. Zhang and L. L. H. Chan, "Overcoming the Challenges of Long-Tail Distribution in Nighttime Vehicle Detection," *IEEE Intelligent Systems*, vol. 39, no. 2, pp. 51–60, 2024, doi: 10.1109/MIS.2024.3350628.

[24] H. Gong and W. Zhuang, "An Improved Method for Extracting Inter-Row Navigation Lines in Nighttime Maize Crops Using YOLOv7-Tiny," *IEEE Access*, vol. 12, pp. 27444–27455, 2024, doi: 10.1109/ACCESS.2024.3365555.

[25] L. Encío *et al.,* "Enhanced Nighttime Vehicle Detection for On-Board Processing," *IEEE Access*, vol. 13, pp. 44817–44835, 2025, doi: 10.1109/ACCESS.2025.3548837.

[26] W. Zhao, T. Wang, A. Tan, and C. Ren, "Nighttime Pedestrian Detection Based on a Fusion of Visual Information and Millimeter-Wave Radar," *IEEE Access*, vol. 11, pp. 68439–68451, 2023, doi: 10.1109/ACCESS.2023.3291398.

[27] Y. Xu, K. Chu, and J. Zhang, "Nighttime Vehicle Detection Algorithm Based on Improved Faster-RCNN," *IEEE Access*, vol. 12, pp. 19299–19306, 2024, doi: 10.1109/ACCESS.2023.3347791.

[28] X. Ma, X. Dai, Y. Bai, Y. Wang, and Y. Fu, "Rewrite the Stars," *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 5694-5703, doi: 10.1109/CVPR52733.2024.00544.

[29] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, 2018, doi: 10.1109/CVPR.2018.00745.

[30] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8759–8768, 2018, doi: 10.1109/CVPR.2018.00913.

[31] Z. Zheng *et al.*, "Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation," *IEEE Transactions on Cybernetics*, vol. 52, no. 8, pp. 8574–8586, 2022, doi: 10.1109/TCYB.2021.3095305.

[32] X. Li *et al.*, "Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection," *arXiv*, 2020, doi: 10.48550/ARXIV.2006.04388.

[33] A. U. Ruby, P. Theerthagiri, I. J. Jacob, and Y. Vamsidhar, "Binary Cross Entropy with Deep Learning Technique for Image Classification," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 4, pp. 5393–5397, 2020, doi: 10.30534/ijatcse/2020/175942020.

[34] A. Vaswani *et al.*, "Attention Is All You Need," *arXiv*, 2017, doi: 10.48550/ARXIV.1706.03762.

[35] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, 2017, doi: 10.1109/CVPR.2017.195.

[36] Y. Xiao *et al.*, "SimpleCNN-UNet: An optic disc image segmentation network based on efficient small-kernel convolutions," *Expert Systems with Applications*, vol. 256, p. 124935, Dec. 2024, doi: 10.1016/j.eswa.2024.124935

[37] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.

[38] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10778–10787, 2020, doi: 10.1109/CVPR42600.2020.01079.

[39] Y. Zhao *et al.*, "DETRs Beat YOLOs on Real-time Object Detection," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16965–16974, 2024, doi: 10.1109/CVPR52733.2024.01605.

[40] Z. Liu *et al.*, "Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9992–10002, 2021, doi: 10.1109/ICCV48922.2021.00986.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[42] C. Li *et al.*, "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," *arXiv*, 2022, doi: 10.48550/ARXIV.2209.02976.

[43] R. Xu, D. Zhu, W. Pang, and M. Chen, "An Underwater Low-Light Image Enhancement Algorithm Based on Image Fusion and Color Balance," *Journal of Marine Science and Engineering*, vol. 13, no. 11, 2025, doi: 10.3390/jmse13112049.

[44] J. Ding, J. Hu, J. Lin, and X. Zhang, "Lightweight enhanced YOLOv8n underwater object detection network for low light environments," *Scientific Reports*, vol. 14, no. 1, Nov. 2024, doi: 10.1038/s41598-024-79211-7.

[45] M. B. Muhammad and M. Yeasin, "Eigen-CAM: Class Activation Map using Principal Components," in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, 2020, doi: 10.1109/IJCNN48605.2020.9206626.

[46] Y. Tian, Q. Ye, and D. Doermann, "YOLOv12: Attention-Centric Real-Time Object Detectors," *arXiv*, 2025.

BIOGRAPHIES OF AUTHORS

Muhamad Dwisnanto Putro received his Bachelor of Engineering (S.T.) degree in Electrical Engineering from Sam Ratulangi University, Manado, Indonesia, in 2010. He earned his Master of Engineering (M.Eng.) degree from the Department of Electrical Engineering, Gadjah Mada University, Yogyakarta, Indonesia, in 2012, and his Ph.D. degree in Electrical, Electronic, and Computer Engineering from the University of Ulsan, South Korea, in 2022. He is currently an Associate Professor in the Magister Program of Informatics at Sam Ratulangi University, where he also serves as the leader of the Algorithmic Intelligence for Vision (AI-VISION) research group. His current research interests include intelligence vision and deep learning, with a particular focus on robotic vision and perception. He can be contacted at email: dwisnantoputro@unsrat.ac.id.

Jane Litouw received a Bachelor's of Engineering (S.T.) in Electrical Engineering from Sam Ratulangi University in Manado, Indonesia, in 2003. He received an Magister Teknik (M.T) degree from STEI ITB in Bandung, Indonesia, in 2014. In 2005 she joined the Department of Electrical Engineering, Sam Ratulangi University, as lecturer. Her current research interests are fuzzy logic system, image processing and deep learning. She can be contacted at email: janelitouw@unsrat.ac.id.

Abdul Haris Ontowirjo received the B.Eng. degree in Electrical Engineering from the Institut Teknologi Bandung (ITB), Bandung, Indonesia, in 1991, M.Eng. degree in Electrical Engineering from the Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia, in 2010. His current research interest includes automation and intelligent control. He can be contacted at email: aharisjo@unsrat.ac.id.

Sherwin Reinaldo U. Sompie received his Bachelor of Engineering (S.T.) degree in Electrical Engineering from Petra Christian University, Surabaya, Indonesia, in 2002, and his Master's degree in Electrical Engineering from Pelita Harapan University, Jakarta, Indonesia, in 2011. He is currently an Assistant Professor in the Department of Electrical Engineering at Sam Ratulangi University, Manado, Indonesia. His current research interests include informatics and artificial intelligence. He can be contacted at email: aldo@unsrat.ac.id.

Hebron Prasetya received his Bachelor of Informatics (S.Kom.) degree from the Informatics Program from Sam Ratulangi University, Manado, Indonesia, in 2025. He is currently pursuing his Master's degree in Informatics at Sam Ratulangi University, where he is also a member of the Algorithmic Intelligence for Vision (AIVISION) research group. His current research interests include computer vision and deep learning. He can be contacted at email: hebronprasetya26@gmail.com.