

Lightweight Enhancement Module for Efficient Sea Turtle Species Detection under Illumination Variation

Yuliana Mose¹, Alex Copernikus Andaria², Hebron Prasetya³, Muhamad Dwisnanto Putro⁴

^{1,2}Department of Computer Engineering, Faculty of Science, Technology and Teaching, Trinita University, Manado, Indonesia

^{3,4}Magister of Informatics, Postgraduate Program, Sam Ratulangi University, Manado, Indonesia

Article Info

Article history:

Received Oct 22, 2025

Revised Mar 23, 2026

Accepted Apr 16, 2026

Keywords:

Sea turtle

Species detection

Illumination

Lightweight Enhancement

Efficient deep learning

ABSTRACT

Sea turtle monitoring is crucial for maintaining ecosystem balance, although traditional methods are costly and inefficient. Object detection provides a foundation for autonomous monitoring by localizing sea turtles in their natural habitat. At the same time, recent deep learning models can operate in real-time. On the other hand, underwater detection remains challenging due to extreme illumination, underwater blur, and underwater distortions, which obscure key features from the background. To address these issues, this work proposes a new sea turtle detector, based on a modified YOLOv12-nano model that incorporates a lightweight enhancement module to improve robustness under diverse illumination. The network introduced Efficient Spatial Convolution (ESC) at the end of the backbone to improve the representation of high-level features. It involves Dual Spatial Convolution (DSC) to improve backbone feature extraction with a large kernel size while preserving efficiency. The Lite module is designed to enrich feature diversity and selectively enhance the quality of high-level features. These modifications significantly enhance detection performance under diverse underwater lighting conditions. The proposed network achieves higher precision than the original YOLOv12-nano while maintaining efficiency, making it well-suited for deployment on low-cost devices.

Copyright © 2026 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Yuliana Mose

Department of Computer Engineering, Faculty of Science, Technology and Teaching, Trinita University
Malalayang, Manado, Indonesia Email: moseyuliana75@gmail.com

1. INTRODUCTION

The presence of sea turtles plays a crucial role both ecologically [1] and economically through their contribution to ecotourism and the blue economy. As iconic marine species, sea turtles attract significant attention from tourists, creating business opportunities for coastal communities in the ecotourism sector. However, human activities, such as illegal hunting, threaten sea turtle populations, disrupting ecosystem balance [2] and having a serious impact on the habitat. Conservation efforts have been carried out through sea turtle population monitoring [3], [4], but general methods are still conventional, such as direct observation, which is often inaccurate, inefficient, and requires significant resources. To enhance conservation effectiveness, an automated vision-based monitoring system can accurately identify sea turtles, using deep learning algorithms.

Deep learning-based underwater animal detection faces several challenges, including extreme illumination variations, depth, and water turbidity. These challenges led to the loss of critical image features and a reduction in model performance. Furthermore, underwater challenges such as noise, color distortion, and reduced visibility were faced in previous studies [5–9]. It also makes it difficult for the model to capture the features needed for accurate detection. In addition, image contrast in underwater environments is degraded

by light scattering and absorption. It leads to missed detections and false positive predictions. Subsequently, underwater detection systems require effective solutions to address these challenges. Such systems support marine life monitoring, ecological research, and autonomous underwater surveillance. To improve the model's robustness, this study employs data augmentation during training. The model is exposed to a variety of simulated underwater conditions to improve its ability to generalize across different environments.

You Only Look Once (YOLO) is a one-stage object detector that performs object classification and bounding box prediction in a single forward pass. This design enables fast and efficient inference. Lightweight YOLO variants further improve computational efficiency, although they often sacrifice detection accuracy. Despite this trade-off, YOLO is widely used as a baseline for underwater object detection because of its high inference speed. Due to illumination variations, a lightweight yet effective YOLO-based model is required to achieve more real-time detection under challenging underwater conditions. This algorithm has presented several versions and variants. The latest iteration, YOLOv12 [10], enhances the YOLO framework by introducing an attention-oriented architecture that enhances detection performance. Several architectural adjustments are introduced to better align attention mechanisms with the YOLO system. It also integrates an attention module to alleviate memory access bottlenecks and removes positional encoding for improved simplicity and speed. It achieves a balanced computational load and uses convolutional operators wherever possible to capitalize on their computational efficiency.

The YOLOv12n offers reduced computational cost and higher speed. However, the structure of YOLOv12 is limited to discriminating complex features. Extensive work is required to enhance YOLOv12n performance in underwater environments. It also enhances detection performance for sea turtle detection under varying illumination. In addition, an efficient model allows the detector to operate on low-cost devices. YOLOv12n combines attention mechanisms with an optimized architecture to improve its suitability for real-time marine conservation. It also preserves the computational efficiency required for practical underwater monitoring applications. This enhancement module captures important features by selectively preserving valuable information while suppressing trivial details. The main contributions of this work are summarized as follows:

1. A new sea turtle detector is introduced to capture the species under varying illumination conditions. The proposed model achieves lower parameters than YOLOv12-nano while maintaining comparable GFLOPs. It ensures efficient operation even on low-power devices.
2. A lightweight enhancement module is introduced to strengthen discrimination performance by highlighting useful information, namely the Lite module. It is integrated to support the central feature extractor. In addition, a modulation InceptionNeXt module is incorporated to emphasize the most relevant features. This approach enhances overall feature representation.
3. A novel lightweight convolution module is proposed to enhance high-level feature extraction, named ESC (Efficient Spatial Convolution). In addition, the proposed Dual Spatial Convolution (DSC) module captures features with larger receptive fields using two different convolution kernels. This design maintains low computational cost.
4. A comprehensive evaluation is performed to examine the performance and efficiency of the proposed lightweight YOLO architecture. In addition, various illumination conditions are considered during the evaluation. The mean Average Precision (mAP) metric is used to evaluate detection performance. Furthermore, the number of parameters, GFLOPs, and inference speed (FPS) metrics are used to evaluate the computational efficiency on CPU-based devices. Comparisons with other lightweight YOLO variants highlight the balance between precision and efficiency achieved by the proposed model.

2. RELATED WORKS

Low illumination poses visual challenges in the underwater environment. Besides, manual observation remains costly and limited. These issues encourage the development of automated detection methods. Therefore, computer vision and deep learning methods are commonly implemented for monitoring systems. Detection by the YOLO algorithm is commonly used for real-time applications. According to one study [11], YOLOv8 was modified using an attention mechanism to resolve these issues. This would be helpful for contour

detection and focusing on the important features of underwater images. In the same way, Underwater Optical Detection Network (UODN) uses efficient feature extraction modules that overcome the problem of image degradation [12]. Recent studies have proposed marine animal detectors that incorporate CARAFE upsampling and a 3D attention mechanism to improve detection performance [13]. Another study introduced a lightweight detector that integrates the C3b and SPPF modules to reduce computational cost while maintaining detection accuracy [14].

Illumination challenges have driven the development of underwater object detectors [15]. A work [16] introduced MAW-YOLOv11 by integrating multi-scale edge selection, image enhancement, and modified feature extractors. The YOLO-GE was proposed by Li et al. [17] and utilizes feature enhancement modules. Besides, underwater-YOLO applies dilated deformable convolutions and a dual-branch occlusion attention mechanism [18]. YOLO has evolved into a widely recognized single-stage detector due to its real-time efficiency. On the other hand, comparative studies have shown that two-stage detectors can achieve competitive performance. An improved network [19] can improve inference speed and mAP and enable comparison with YOLOv3, YOLOv8, and Faster R-CNN for marine object detection. Another study applied Faster R-CNN to underwater human detection. It can highlight the potential for safety-critical applications [20]. These studies incur high computational costs and reduce speed in real-time systems. In order to address this limitation, LFN-YOLO uses reparameterization and SPD Conv to reduce computational complexity [21].

The development of sea turtle detection remains limited, presenting significant research opportunities. Early studies using the HOG method [22] achieved low accuracy. In contrast, the application of ResNet reported high performance according to standard object detection evaluation metrics. Furthermore, another study [23] employing a Graph Convolutional Network (GCN) demonstrated robust and flexible performance. A bibliometric analysis can identify potential future research directions [24]. However, these studies were limited to object classification. They cannot determine object locations, which is essential for enabling more effective detection, tracking, and behavioral analysis. Efficient and detailed conservation strategies. To overcome these challenges, some researchers have developed more sophisticated techniques for object detection. For example, Noguchi et al. [3] combined YOLOv7 with BoT-SORT to detect and count green sea turtles in drone imagery. Another study [25] extended YOLOv8 by incorporating the Excited Group Attention (EGA) module. This model created feature extraction even on limited resources. Another study [26] introduced YOLOv8p for sea turtle detection. In addition, [27] proposed a CNN-based model to estimate sea turtle populations from drone imagery. Another study [28] focused solely on sea turtle image classification.

In this work, the proposed architecture is inspired by YOLOv12 [10], which is designed as a lightweight framework that consist of four primary components: a feature extractor, attention modules, a path aggregation network, and detection layers. Specifically, we adopt and modify the nano version of YOLOv12 to enhance inference speed while maintaining detection accuracy. To achieve this objective, the backbone efficiency is optimized by reducing convolutional channels and integrating Dual Spatial Convolutional and Efficient Spatial Convolutional modules. These improvements enable a more computationally efficient model while preserving the precision of the sea turtle location.

3. METHOD

The quality of essential feature is a key factor in the information extraction process. The discrimination stage keeps important information and removes irrelevant details. As illustrate in Figure 1, The proposed module extractor progressively constructs representations from simple to complex patterns, enabling the architecture to develop a deeper understanding of the input data. Each layer in the backbone progressively constructs a hierarchical structure, starting from low-level patterns to high-level features relevant to the object information. Therefore, the backbone is a crucial component of the architecture that influences the system's performance.

3.1. Backbone

The proposed backbone adopts the YOLOv12 structure by employing the main extractor modules, such as A2C2f, C3K2, and Conv. The Conv module involves 3×3 convolutions with a stride of two, followed by batch normalization and SiLU activation. This mechanism can effectively reduce the dimensions of the feature map compared to the pooling operation. However, it still lacks extraction effectiveness, where the several-layer demands a higher computational cost. To enhance both the performance and efficiency of feature extraction, several modifications are incorporated into the proposed network. A2C2f and C3K2 modules remain with significant transformations at the eighth layer. The strategy replaces the A2C2f block in the last backbone

with a modified C3K2 block and modified inception. The new module, called Efficient Spatial Convolution (ESC), substitutes the bottleneck module. This replacement is utilized to overcome the limitations of the A2C2f module in capturing essential information. Furthermore, a Dual Spatial Convolutional module is incorporated to enrich spatial representation.

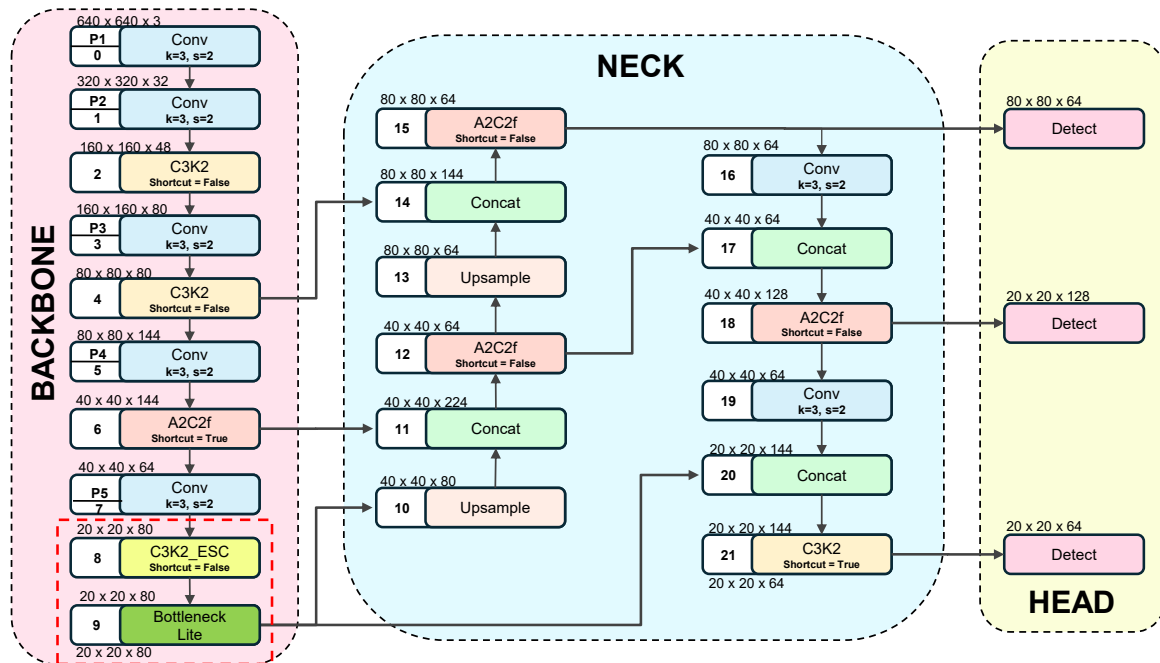


Figure 1. The general architecture of turtle detection. The proposed Lightweight attention is inserted on the backbone's last part. The area marked in red indicates the modification.

3.1.1. A2C2f

In complex visual environments, extracting meaningful information from feature maps is crucial for accurate object detection, yet standard convolutions are often insufficient. To address this, YOLOv12 introduces A2C2f, also known as Area Attention with C2f, which builds on the C2f structure. In the YOLOv12 turbo version, this enhancement module uses Flash Attention, which is computationally lighter than standard Area Attention. It begins with a 1×1 convolution that reduces the size of channels to half, which simplifies computation while preserving key information and improving the relationship between feature maps. The module then splits into two branches. One acts as a bypass, while the other applies either Area Attention with Feed Forward layers or C3K modules. Area Attention works similarly to self-attention in YOLOv10 and YOLOv11 but divides the feature map into smaller areas, making faster matrix multiplication operations. It utilizes two branches, one applies a 1×1 convolution to double the channels and generate the Query and Key. Moreover, the second branch also employs a 1×1 convolution to create the Value representation for capturing feature relationships. The Query, Key, and Value maps reshape into matrices. Each matrix is divided into smaller areas to enable batch processing. Furthermore, Query and Key are separate to assign their roles in the attention mechanism. Matrix multiplication between Query and Key highlights the most relevant positions, which are then scaled and normalized with softmax to recalibrate the Value feature maps. In parallel, the other Value branch involves a 5×5 depthwise convolution that learns dynamic weights. The neuron weights can enhance the Value information by emphasizing valuable regions. Furthermore, a Feed Forward layer refines the features using two 1×1 convolutions. The first expands the channels with batch normalization and SiLU activation. The second restores the original size without activation. Both Area Attention and Feed Forward involve residual connections that can be stacked multiple times. In order to obtain richer feature representations, it fuses the outputs of two important blocks. Finally, it uses a 1×1 convolution to integrate these features and reconstruct the channel dimension.

3.1.2. C3K2

Underwater conditions present several challenges for computer vision, including blur, low illumination, and complex backgrounds. To overcome these challenges, multiple convolutional layers are required to extract robust features under varying conditions. The C3K2 module is one of the main feature extractors, which is designed as a faster implementation of the CSP Bottleneck with two convolutions. This module is similar to C2f (Convolutional Two Faster) but uses the C3K module. It contains several convolutional layers, first applying 1×1 convolution to adjust the channel size and enable information sharing across them. For efficiency, only half of the channels are used for feature extraction, while the other half is employed as an identity value. C3K2 has two selectable modes, allowing the use of the C3K module with three convolutions and an adjustable kernel size bottleneck or the C2f structure with two 3×3 convolutions for feature extraction. Since C3K2 has the same structure as C2f, it combines the outputs from each C3K process or from the bottleneck if disabled using a concatenation operation to enrich the feature representation. The other half of the channels is also combined with the processed features from C3K. Additionally, a 1×1 convolution is applied to capture relationships between pixels across channels. This results in a rich and well-extracted set of features.

3.1.3. Efficient Spatial Convolution

Feature extraction is essential for capturing key information. Object detection networks involve a sequence of convolutional layers that work together to find distinctive features. The proposed architecture introduces an Efficient Spatial Convolution (ESC) module that integrates Dual Spatial Convolution (DSC) to improve backbone ability discrimination. This module is placed at the backbone to strengthen the spatial relationships in high-level feature maps. To improve performance efficiency, the ESC module employs splitting channels into G_A , and G_B . This technique applies two distinct processes, allowing for the lightweight handling of relevant information. As shown in Figure 2, the first process applies a 1×1 convolution to reconstruct feature maps and enable information flow between channels. It then uses a Dual Spatial Convolution to enhance extraction and strengthen spatial interactions. Layer normalization is also involved to stabilize variation and capture relationships across channels and spatial dimensions within each batch. In order to refine the input feature, this module utilizes a residual strategy. The process can be described as follows:

$$ESC = LayerNorm(Concat[K_1, K_2]) + x. \quad (1)$$

The process is divided into two segments into K_1 , and K_2 . Each part receives half of the input channels to pursue the computation process efficiently. The K_1 segment applies Dual Spatial Convolution (DSC) twice using two blocks. This block combines depthwise convolution and axial convolution with a large kernel size while maintaining efficiency. In addition, the other branch is a 1×1 convolution to reconstruct the original information. This module process can be expressed as follows:

$$K_1 = DSC(DSC(G_B)), K_2 = CV_{1 \times 1}(G_A), \quad (2)$$

where

$$[G_A, G_B] = x. \quad (3)$$

Large kernels are capable of capturing richer information across spatial features. However, they require more neuron parameters and computation cost. To address this issue, depthwise convolution is employed to extract spatial information without mixing the channel information. Additionally, ESC involves a 7×7 depthwise convolution and axial convolutions, which avoids the network's higher computational cost. Meanwhile, the other ESC branch uses a 1×1 convolution ($CV_{1 \times 1}$) followed by batch normalization and SiLU activation to strengthen channel relationships. The K_1 branch focuses on extracting spatial information and ignores the channel interaction. Therefore, the K_2 branch employs a 1×1 convolution to merge channel information. Moreover, each output features are combined through an addition operation to refine the overall feature representation. The proposed model integrates spatial feature representation with efficient channel extraction to enhance feature discrimination, thus improving the network's ability to accurately separate turtle features under various lighting conditions.

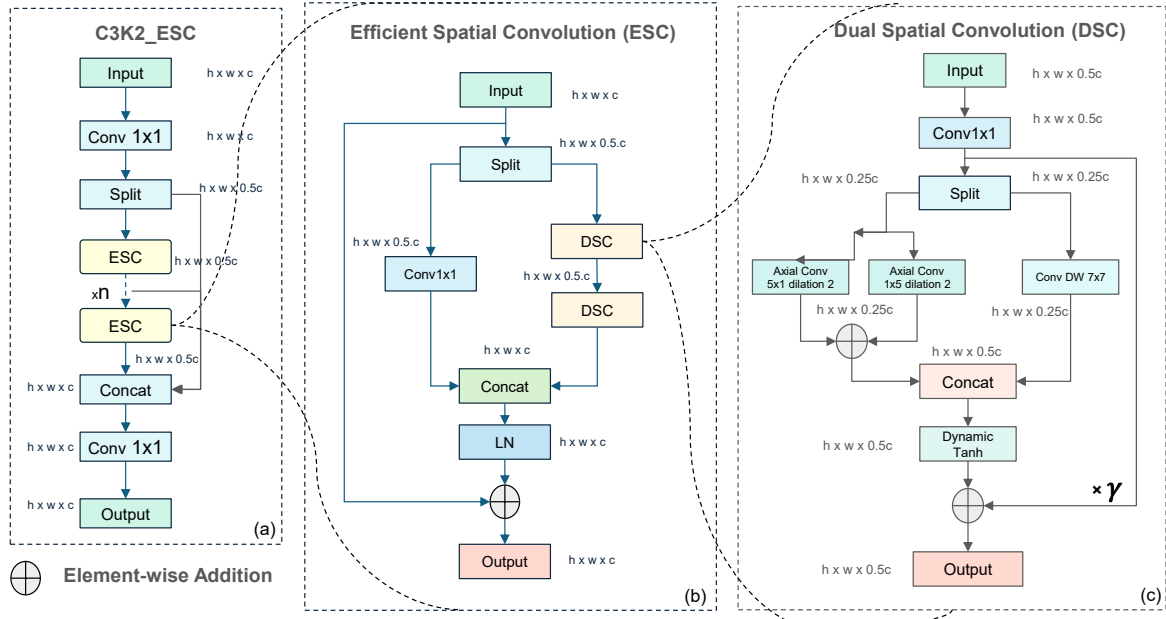


Figure 2. Structural illustration of the proposed lightweight convolutional modules, including a). C3K2_ESC, b). Efficient Spatial Convolution, and c). Dual Spatial Convolution.

3.1.4. Dual Spatial Convolution

Local spatial information encodes object details that are critical in distinguishing the target from its surrounding background. To effectively discriminate these features, convolution operations with weighted filters are employed. However, a 3×3 convolution has a limited receptive field, which restricts its ability to capture broader contextual information. Increasing the kernel size can enlarge the receptive field and improve local feature extraction. However, large convolution kernels also increase the number of parameters and computational cost. To address this issue, the proposed DSC block combines large-kernel convolutions with depthwise and axial convolutions. As shown in Fig. 2, this design enhances spatial feature representation while maintaining computational efficiency. On the other hand, the axial path highlights the directionality of the convolution operation by splitting it into horizontal and vertical. This combination of the two different types of representation within the DSC layer helps keep rich object information while modeling spatial relations. In addition, the DSC adopts a two-branch design where the input channels of the ESC are divided into two branches, with each branch receiving 25% of the input channels. The DSC process may be depicted as shown below:

$$DSC = DyT(Concat[RA, RB]) + Gamma(C_1). \quad (4)$$

The RA and RB have been formulated into two parallel paths for ease of computation. This is achieved through a channel-splitting operation. Half of the input feature maps from the ESC module are processed in parallel, which improves computational efficiency while preserving feature diversity. After this, the input features are improved by applying a residual function. The input features are then refined using a residual connection. A learnable scaling parameter, Γ , is applied to the refined feature maps to preserve useful information. Moreover, Dynamic Tanh [29] is applied after merging the two feature representations. It normalizes the fused features, improving feature stability and the robustness of the combined representation. In this way, the DSC module allows the network to better recognize the channels with the relevant information about the objects. The RA module uses the large-depthwise convolution along with the single-kernel convolution to learn the extensive spatial relations and retain the local feature details in this manner:

$$RA = CV_{1 \times 1}(CDW_{7 \times 7}(C_A)). \quad (5)$$

This method uses depthwise convolution using a 7×7 filter ($CDW_{7 \times 7}$). Compared with conventional convolution, this method requires fewer parameters and lower computational cost. However, this method has a limited ability to model inter-channel relationships. This means that an additional convolution is needed

to model inter-channel relations. Similar to the depthwise separable convolution, this method employs a 1×1 convolution $CV_{1 \times 1}$. In addition, the second layer uses axial convolutions, in which the process is split into horizontal and vertical components. In this part, two axial processes are combined to form a single feature map that can be represented as:

$$RB = AC_{1 \times 5}(C_B) + AC_{5 \times 1}(C_B). \quad (6)$$

Axial convolution is widely used in efficient network architectures because it operates on features along a single dimension, either vertically or horizontally, thereby reducing computational cost without sacrificing feature interactions. The only problem is that it does not capture all the necessary spatial relations. To address this limitation, vertical and horizontal convolutions, $AC_{5 \times 1}$ and $AC_{1 \times 5}$, are employed. Their outputs are then combined through element-wise summation. This will allow the module to capture a broader spatial structure through modeling of vertical and horizontal dependency together. These two branches C_A and C_B are defined from the 1×1 convolution for the reconstruction of the interaction between the feature maps that can be expressed as:

$$[C_A, C_B] = C_1, \quad (7)$$

where

$$C_1 = CV_{1 \times 1}(x). \quad (8)$$

3.1.5. Bottleneck with Lite module

The computation process in convolutions is based on the use of parameters and computations, the quantity of which increases as the kernel size grows or more channels are used. The bottleneck in the C3K2 block solves the problem of increasing the receptive field in an efficient way by using two 3×3 convolutions. This approach allows an effective receptive field equivalent to 5×5 to be obtained. At the same time, it is still not the most efficient approach in terms of kernel usage, since using two convolutions requires twice as much computation as using a 5×5 kernel. This problem is addressed by the bottleneck, which reduces the number of channels in the first convolutional layer to half of the total. An important disadvantage of reducing the number of channels is the decrease in feature extraction efficiency. Conversely, bottleneck structures must capture large receptive fields to strengthen long-range feature correlations. The proposed network presents a Bottleneck with Lite module to capture a wide area of feature maps, and incorporates Light Enhancement Module (Lite) to support the relational feature with efficient approach. This process can be described as follows:

$$Bl = Lite(C_{3 \times 3}(C_{3 \times 3}(x))) + x. \quad (9)$$

This bottleneck employs two $C_{3 \times 3}$ layers, defined as 3×3 convolutions, arranged sequentially to enlarge the effective receptive field while simultaneously reducing computational cost. This design uses approximately 64 % fewer parameters compared to a standard 5×5 convolution. Moreover, Bl incorporates the Lite structure, which combines axial convolution and depthwise convolution to capture broader spatial relationships. It is assigned at the end of the bottleneck in order to perceive the relationship from the feature maps output with a larger area receptive field. It also strengthens the relevant feature from the previous convolution process, which focuses on sea turtle information. A skip connection operation refines the extracted features but also preserves original information, thereby reducing the risk of incorrect weighting. The proposed block incorporates a lightweight module at the beginning of the bottleneck as an initial extractor, enhancing feature extraction by introducing relational variations and feed-forward reconstruction.

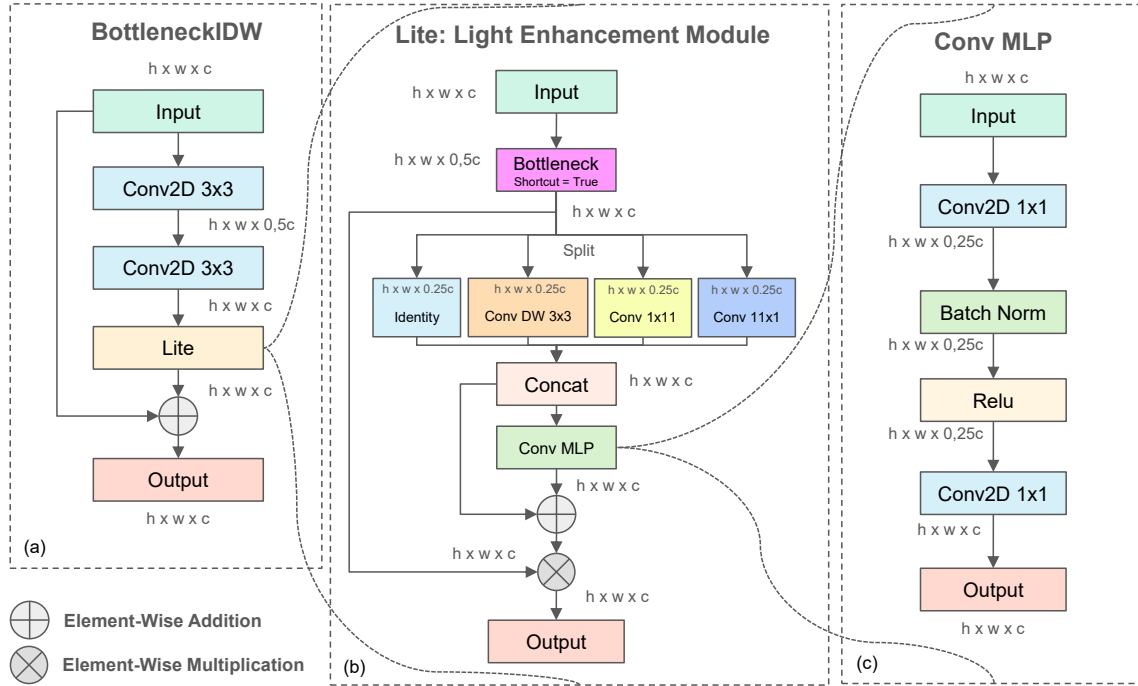


Figure 3. The proposed BottleneckLite module to improve feature extraction efficiency

3.1.6. Lite Module

The backbone functions as the primary feature extractor, constructing a hierarchical pyramid structure from bottom to top that reflects the transition from low-level to high-level features. While low-level features capture fine spatial details, high-level features provide semantically rich representations that emphasize the most relevant regions of interest. These layers are critical for achieving accurate predictions. Therefore, an effective weighting process is employed to enable the model to evaluate and refine information across layers. However, high-level features are sensitive to parameter growth, since they are already derived from multiple convolutional operations. To address this issue, the proposed Lite module incorporates an efficient strategy in the high-level layers by employing depthwise and axial convolutions to maintain parameter efficiency, as shown in Figure 3. In addition, pointwise convolution is utilized to compensate for the limitations of depthwise convolution, ensuring balanced feature extraction and representation. This process can be expressed as follows:

$$Lite = MLP(FS(F_A))(B(x))) * F_B. \quad (10)$$

The Lite module employs multiple branches and utilizes a split operation to reduce computational cost. It employs element-wise multiplication $*$ to refine the input feature maps with the contextual map. In the first stage, Lite applies a bottleneck structure consisting of two 3×3 convolutions, $C_{3 \times 3}$. Each convolution block incorporates batch normalization and SiLU. The bottleneck design further helps minimize parameter usage and maintain computational efficiency. This bottleneck process can be described as follows:

$$B = C_{3 \times 3}(C_{3 \times 3}(x)) + x. \quad (11)$$

At the first stage, the bottleneck module B functions as an efficient feature extractor while simultaneously acting as a selective layer for high-level features with complex representations. By reducing the kernel count to half of the input channels, the module allows the network to emphasize the most salient features through optimized weight distribution, thereby improving representational efficiency without introducing significant computational overhead. In this work, each $C_{3 \times 3}$ convolution is followed by batch normalization and SiLU activation to stabilize feature variations and introduce non-linearity. After the bottleneck process, the feature maps are divided into two branches, which can be described as follows:

$$[F_A, F_B] = P. \quad (12)$$

This splitting process enables the model to preserve the identity feature F_B while simultaneously deriving contextual information from FA. Specifically, F_B functions as the identity pathway, ensuring feature continuity, whereas FA is divided into four parallel branches, each allocated 25% of the total channels. These branches are designed to capture complementary information through depthwise and axial convolutions, thereby enriching the overall representation. The complete process can be expressed as follows:

$$FS = \text{Concat}[S_1, CDW_{3 \times 3}(S_2), AC_{1 \times 11}(S_3), AC_{11 \times 1}(S_4)], \quad (13)$$

where

$$[S_1, S_2, S_3, S_4] = F_A. \quad (14)$$

This part is divided into four sub-branches. The S_1 branch provides the identity path and does not perform any operation. The S_2 branch applies a depthwise convolution $CDW_{3 \times 3}$ with a 3×3 kernel to capture only spatial information. Furthermore, the S_3 and S_4 sub-branches decompose large kernel convolutions into more efficient operations to optimize computational performance. The mechanism is inspired by the Inception module and depthwise convolution, designed to enhance feature diversity while maintaining efficiency and compatibility with various devices, without requiring specialized optimization. Branches S_3 and S_4 perform independent filtering operations using depthwise convolutions to capture broader receptive fields. Specifically, $AC_{1 \times 11}$ refers to a 1×11 convolutional filter applied along the horizontal axis, while $AC_{11 \times 1}$ denotes an 11×1 filter applied along the vertical axis. These complementary operations enable the model to capture elongated spatial dependencies in both directions without incurring the high computational cost of large conventional kernels. Instead of optimizing the kernel size, a large kernel is directly employed to cover a wider receptive field. These branches operate collaboratively to enrich object representation, and their outputs are combined into a single tensor through concatenation, ensuring that the most informative features are preserved and integrated. Moreover, the feature maps are processed with an MLP block to strengthen the feature relationship between channels. It also allows for maintaining the data variation and creating non-linearity. This process can be described as follows:

$$MLP = C_{1 \times 1}(\text{ReLU}(\text{BatchNorm}(C_{1 \times 1}(FS)))) + FS. \quad (15)$$

The MLP process combines information from the four branches to reconstruct relationships between the feature maps. In the first stage, a $C_{1 \times 1}$ filter reduces the channels to 25% of their original size, enabling lightweight computation, while batch normalization stabilizes the data variation. Since feature maps have already applied multiple convolutional transformations, this stage prioritizes the most salient responses while suppressing irrelevant activations. The ReLU activation function eliminates negative values, thereby refining the representation and retaining only meaningful features. Subsequently, a 1×1 convolution ($C_{1 \times 1}$) is applied to restore the channel dimensionality to its default size. Finally, a residual connection is performed, which integrates the refined features with the original input, enhancing representational capacity while stabilizing the learning process.

3.2. Neck

The neck combines information from multi-level features in the backbone. Feature fusion strengthens the connections between feature elements at different stages of convolution. This process applies up-sampling and down-sampling to standardize the dimensions of the feature maps. YOLOv12 utilizes a Path Aggregation Network (PAN) and Feature Pyramid Network (FPN), which generates three hierarchical merging paths to capture bottom-up and top-down information. This model employs a C3K2 extractor to enhance efficiency and accelerate computation. The nearest neighbor method is utilized to increase the matrix dimensions of the feature map, leading to an enlarged feature map size in the detection layer. On the other hand, there are two 3×3 convolutional blocks with stride of two that downsample the feature map into medium and small sized feature maps. Information is interconnected with the features extracted from the backbone utilizing a concatenation operation, which strengthens the combined features to the detection layer.

3.3. Head

YOLOv12 is built on the architecture of YOLOv8, utilizing the same head as an anchor-free model with a decoupled head [26]. It incorporates two branch blocks, each using two 3×3 convolutional layers, followed by a 1×1 convolutional layer, which are applied sequentially on each branch. This design implements

three detection layers with feature map scale assignments of 20×20 , 40×40 , and 80×80 , respectively [27]. The large feature map detection layer, which is 80×80 , detects small objects. The medium feature map detection layer, which is 40×40 , detects medium objects, and the small feature map detection layer, which is 20×20 , detects small objects. Additionally, this design enables the model to process the tasks of objectness, classification, and regression separately, thereby improving overall model performance and efficiency. Similar to YOLOv8, YOLOv12 incorporates loss functions of DFL (Distributional Focal Loss) and CIoU (Complete Intersection over Union) loss for bounding box loss, followed by BCE (Binary Cross Entropy) loss for classification loss (Cls) [30–32]. The overall loss equation used in this work can be expressed as follows:

$$Loss_{total} = \lambda_{CIoU} L_{CIoU} + \lambda_{DFL} L_{DFL} + \lambda_{Cls} L_{Cls}. \quad (16)$$

where λ is a balance parameter for each loss. The default configuration of this balance parameter is set to 7.5, 0.5, and 1.5 for box, class, and DFL loss weights, respectively [1].

3.4. Dataset

In this work, the sea turtle detection dataset proposed in [26] is used for the experiment. The dataset contains three sea turtle species collected from Bunaken National Park. Three different turtles, namely, Hawksbill turtle, Olive Ridley turtle, and Green turtle, are present in the dataset. The images in the dataset are augmented via the illumination adjustment procedure. It will help increase the effectiveness of the model in various underwater lighting conditions. The brightness ($\alpha \in [0.3, 1.2]$) and contrast shift ($\beta \in [-15, 22]$) are applied to the pixel intensity.



Figure 4. Sample images of the dataset showing variations in underwater conditions.

Furthermore, the dataset is divided into three subsets for training, validation, and testing. It contains a total of 48,216 images. Of these, 30,934 images are used for training, 7,726 for validation, and 9,612 for testing. This data partition enables the model to be trained, validated, and evaluated on separate image sets. As shown in Fig. 5, the dataset includes three sea turtle species. These consist of *Chelonia mydas* (Green Turtle) with 28,053 images, *Eretmochelys imbricata* (Hawksbill Turtle) with 11,354 images, and *Lepidochelys olivacea* (Olive Ridley Turtle) with 8,809 images. The distribution provides a diverse set of samples for evaluating the model across different sea turtle species.

3.5. Training Configuration

Experiments are performed on images resized to 640×640 pixels. The model parameters are optimized using the SGD optimizer over 150 epochs to ensure stable convergence. The batch size of 16 and the learning rate of 0.01 are chosen to ensure fast and stable convergence. The experiments were conducted on a PC running Ubuntu 22.04 using the PyTorch framework with CUDA 12.2 acceleration. This configuration reduced the training time. The machine has an Intel Core i3-6100 processor (3.70 GHz), 16 GB of memory, and an NVIDIA GeForce RTX 3060 graphics card with 12 GB of memory. Model performance is evaluated using the mean Average Precision (mAP) metric. The mAP is computed over Intersection over Union (IoU) thresholds ranging from 0.5 to 0.95.

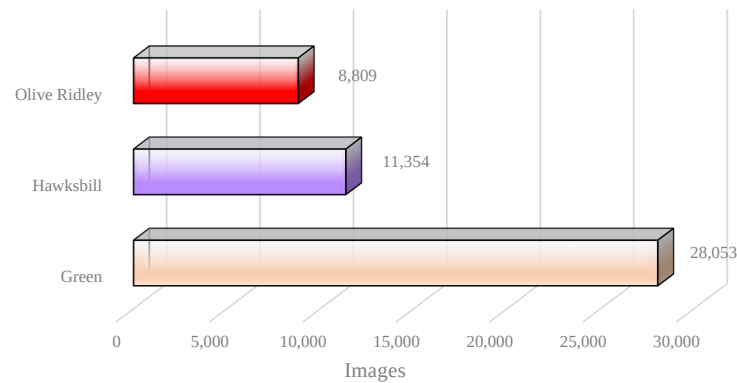


Figure 5. Visualization of image distribution across three sea turtle classes: Green, Hawksbill, and Olive Ridley.

3.6. Inference Configuration

The inference experiments were conducted on a desktop PC with an Intel Core i7-12700 processor (20 threads), 16 GB of RAM, and Ubuntu Linux. This hardware provides sufficient computational capability for inference. In addition, the use of Ubuntu allows the computer to be compatible with multiple machine learning frameworks. It makes the computer efficient in carrying out tasks at a high speed and low latency. The problem of overlapping objects is handled by using the non-maximum suppression (NMS) technique with an Intersection over Union (IoU) threshold of 0.5 and a confidence threshold of 0.25 with 300 detections per frame [33].

Table 1. Analysis of the proposed model in detection performance and complexity cost.

Model	Parameter	GFLOPs	Val		Test	
			mAP50	mAP95	mAP50	mAP95
YOLOv12n	2,520,429	6.0	0.813	0.708	0.811	0.705
YOLOv12h	983,337	5.6	0.824	0.713	0.815	0.709
YOLOv12h+ESC	916,509	5.6	0.826	0.714	0.823	0.712
YOLOv12h+ESC+Lite	1,481,293	6.0	0.845	0.729	0.835	0.724

4. RESULTS AND DISCUSSIONS

4.1. Ablation Study

The deep learning model analysis evaluates the contribution of each module to detection performance and computational cost. The designed modules have been applied to the neural network with consideration for improvements in localization precision, the number of parameters, and computational cost, as shown in Table 1. In the first stage, YOLOv12-nano, as the baseline, has been analyzed. It produces 2,520,492 parameters and requires 6.0 GFLOPs. The proposed model achieved an mAP95 of 0.705 on the test subset. Subsequently, the convolution channels were optimized to improve model performance. The resulting variant is referred to as YOLOv12-hypernano. This model had 4% higher mAP95 than the baseline, 1.5M fewer parameters, and 0.4 fewer GFLOPs. It delivers 3% performance improvement across IoU thresholds ranging from 0.5 to 0.95. In addition, the proposed detector incorporates the Lite module, which enhances performance by 12% through the enhancement of critical feature representations. The complete model maintains computational efficiency with only 1,481,293 parameters and 6.0 GFLOPs, which is competitive with other lightweight object detection networks. Overall, the experimental analysis demonstrates that the proposed modules effectively improve detection performance while preserving efficiency. Consequently, the detector offers substantial advantages in optimization, achieving a balanced trade-off between accurate turtle localization under varying illumination conditions and minimized computational and parameter requirements.

4.2. Evaluation Dataset

Table 2. Performance comparison of YOLO variants on validation and test datasets, including number of parameters and computational complexity cost.

Model	Parameter	GFlops	Val		Test	
			mAP50	mAP95	mAP50	mAP95
YOLOv12n	2,520,429	6.0	0.813	0.708	0.811	0.705
YOLOv11n	2,592,729	6.4	0.814	0.712	0.809	0.708
YOLOv10n	2,708,210	8.4	0.813	0.712	0.815	0.716
YOLOv8n	3,011,433	9.2	0.829	0.722	0.830	0.721
YOLOv5n	2,509,049	7.2	0.821	0.720	0.818	0.712
YOLOv12h	983,337	5.6	0.824	0.713	0.815	0.709
YOLOv12h+ESC	916,509	5.6	0.826	0.714	0.823	0.712
YOLOv12h+ESC+SE	1,359,853	5.9	0.824	0.715	0.814	0.711
YOLOv12h+ESC+ECA	1,358,259	5.9	0.821	0.722	0.818	0.719
YOLOv12h+ESC+CBAM	1,360,049	5.9	0.818	0.717	0.809	0.712
YOLOv12h+ESC+Bilinear Mapping	1,431,181	6.0	0.81	0.702	0.801	0.698
YOLOv12h+ESC+Efficient Modulation	1,495,693	6.0	0.817	0.717	0.807	0.708
YOLOv12h+ESC+Transformer	1,415,053	6.0	0.800	0.700	0.800	0.700
Proposed	1,481,293	6.0	0.845	0.729	0.835	0.724

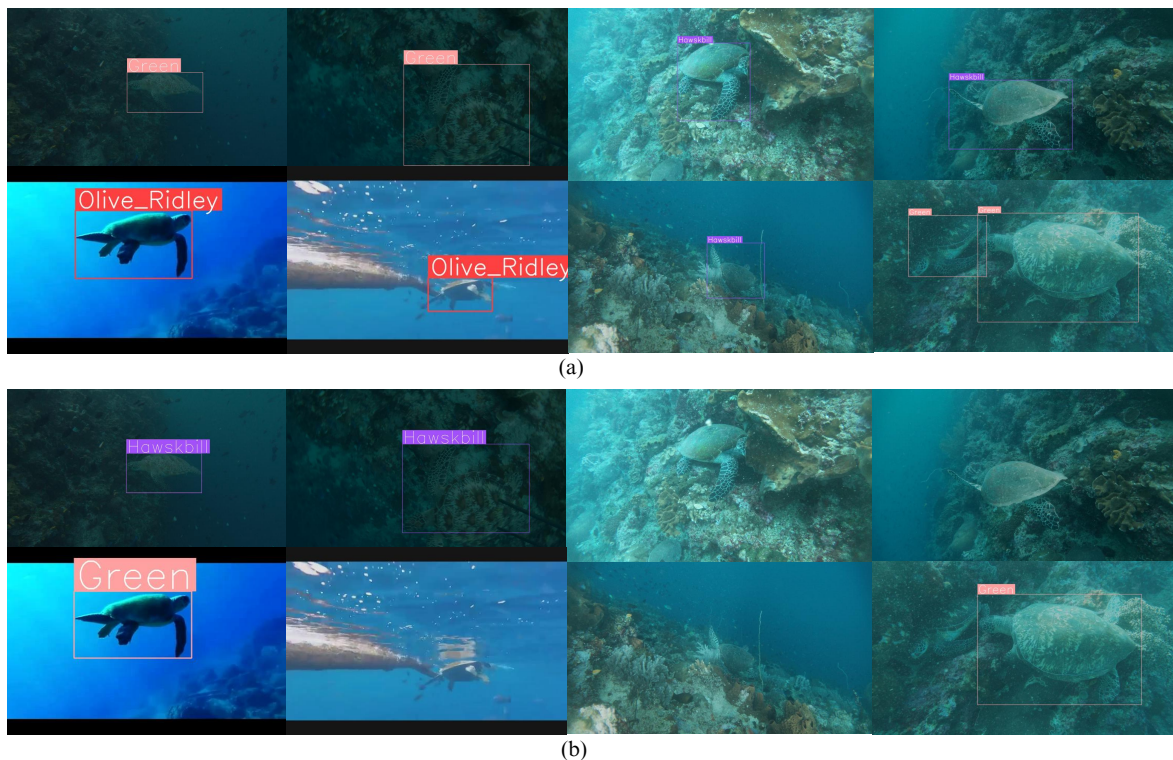


Figure 6. Visualization of detection results for different sea turtle species, including Green, Hawksbill, and Olive Ridley, using (a) the proposed model and (b) the original YOLOv12-nano.

According to the Table 2., the performance and efficiency of each module are evaluated. The table presents a comparison between the proposed module and several YOLO variants. To ensure fairness, only lightweight versions of the YOLO family are included. All models are evaluated on the same dataset to maintain consistency. The earlier generation YOLOv5-nano requires 2.5 million parameters and 7.2 GFLOPs to achieve an mAP95 of 0.712. Moreover, YOLOv8-nano achieves an mAP95 of 0.721 but requires 3.01 million parameters and 9.2 GFLOPs, which is approximately 500K more parameters than YOLOv5-nano. In addition, YOLOv10-nano requires 2.7 million parameters and 8.4 GFLOPs. However, its performance decreases by about 0.5% compared to YOLOv8-nano, achieving an mAP95 of 0.716. Interestingly, several of the latest YOLO architectures, such as YOLOv11-nano and YOLOv12-nano, also show lower accuracy than earlier variants, with mAP95 scores of 0.708 and 0.705, respectively. Nevertheless, both models maintain high

computational efficiency and are well-suited for deployment in resource-constrained environments.

In this study, the proposed model is developed by modifying YOLOv12-nano, which is selected as the base architecture for its potential to enhance performance while preserving efficiency through the integration of an enhancement module. The modification process begins by reducing the number of channel configurations, as illustrated in Figure 1, resulting in a new variant referred to as YOLOv12h (Hyper Nano). This variant contains 983 thousand parameters, representing a 61% reduction relative to the original model. Additionally, the computational cost decreases to 5.6 GFLOPs while maintaining a competitive mAP95 of 0.709. This approach helps to allocate some space for the parameters and GFLOPs. It creates an opportunity to apply additional enhancement modules to improve the detection performance without significantly increasing the computational complexity.

Furthermore, experiments show that YOLOv12h equipped with Efficient Spatial Convolution (ESC) achieves higher performance with an mAP95 of 0.712. This modification results in a 6.8% reduction in parameters compared to the baseline YOLOv12h, primarily due to replacing several standard convolution layers with lightweight alternatives. To further assess the proposed Lite module, additional attention mechanisms were integrated for comparison. The combination of Squeeze-and-Excitation (SE) module with YOLOv12h+ESC slightly decreases performance to an mAP95 of 0.711 while increasing both parameters and GFLOPs to 1.35 million and 5.9, respectively. On the other hand, the Efficient Channel Attention (ECA) module improves performance to mAP95 of 0.719 with a similar parameter cost. Subsequently, the Convolutional Block Attention Module (CBAM) generates 1.36 million parameters with 5.9 GFLOPs to achieve mAP95 of 0.712, which is slightly lower than the ECA module despite having a higher parameter cost. Furthermore, the integration of the Bilinear Mapping module [34], one of the current attention mechanisms, achieves an mAP50:95 of 0.698 with 1,431,181 parameters and 6.0 GFLOPs. Meanwhile, the Efficient Modulation [35] achieves improved accuracy with an mAP50:95 of 0.708 and 1,495,693 parameters under the same 6.0 GFLOPs computational budget.

The comparison also evaluates area attention in transformers replacing the Lite module, which performs worse than other attention mechanisms. This variant achieves an mAP95 of 0.7 while consuming more resources, with 1.4 million parameters and 6 GFLOPs. In contrast, the proposed model delivers the best overall results, attaining an mAP95 of 0.724 with 1.48 million parameters and 6 GFLOPs. Compared to YOLOv12-nano, it not only achieves higher accuracy but also maintains lower computational cost, confirming its superiority over other architectures.

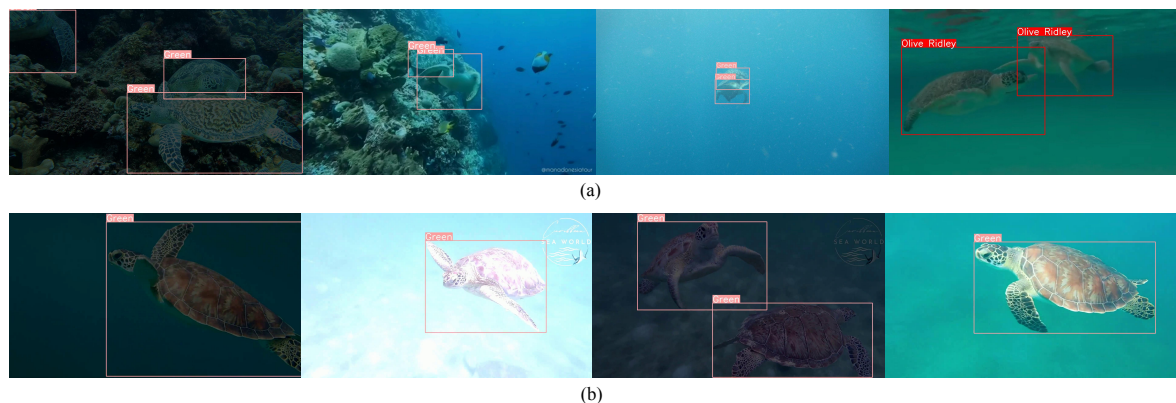


Figure 7. Visualization of detection results for sea turtles: (a) overlapping scenarios with partial occlusion, and (b) varying underwater conditions in the Caribbean Sea.

In order to evaluate the performance of the proposed model, Figure 6 compares the proposed architecture (a) with the original YOLOv12-nano (b). The original model struggles in complex backgrounds where sea turtle features are less distinguishable, causing misclassification between Green and Hawksbill turtles due to their similar patterns. ESC and DSC enable the model to handle these challenges more effectively by enhancing local feature extraction and capturing fine details such as shell textures and edge patterns. These modules strengthen the model's ability to separate subtle inter-class differences while maintaining robust recognition in cluttered scenes.

As a result, the overall detection pipeline benefits from higher mAP scores and more precise localization of sea turtles. In addition, the comparison highlights how the proposed model maintains stable detection

across varying illumination conditions. While the original YOLOv12-nano struggles to retain consistent confidence scores in darker or low-contrast regions, the proposed model adapts more effectively, ensuring that bounding boxes remain accurate and tightly aligned with the turtles. This consistency is particularly critical for reliable monitoring in marine environments, where visibility is frequently affected by fluctuating illumination. In order to evaluate model robustness, Figure 7 presents detection results under overlapping conditions and in various marine environments. Figure 7 (a) shows detection results in Asian waters, illustrating performance in scenarios with overlapping objects where sea turtles are partially occluded. Despite this challenge, the proposed model effectively distinguishes between different instances, even when they appear in similar positions and align with the camera view. Figure 7 (b) presents detection results under varying water conditions in the Caribbean Sea, including differences in color, illumination, and visibility. The results demonstrate that the proposed model remains robust and can accurately detect sea turtles under these environmental variations.

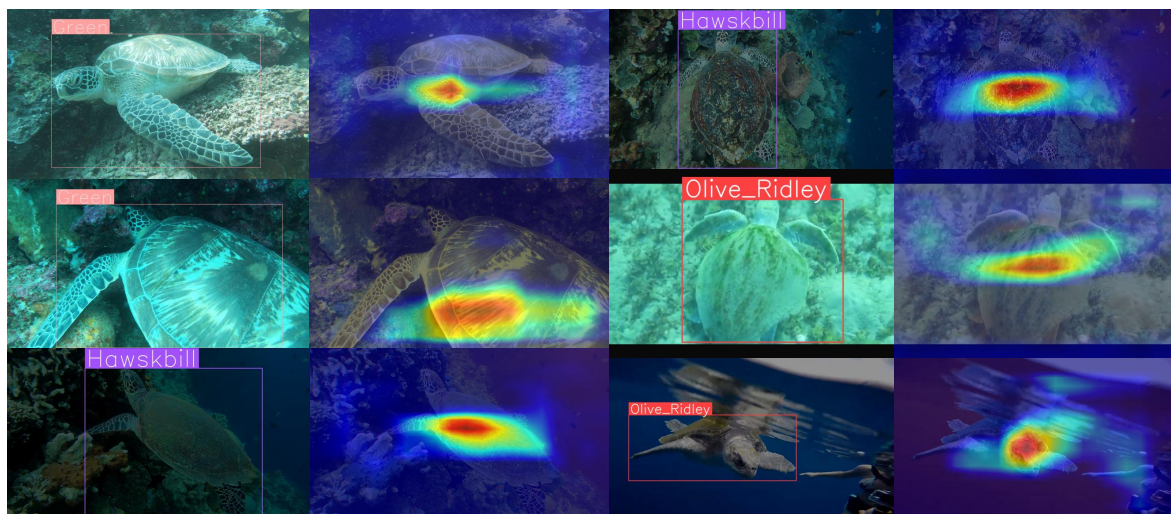


Figure 8. Visualization of detection results with corresponding heatmaps for different sea turtle species (Green, Hawksbill, and Olive Ridley) under varying underwater illumination conditions.

4.3. Heatmap Evaluation

Figure 8 shows the heatmaps, which were created using EigenCAM [36]. This observation provides insight into how the model identifies the essential features of the sea turtle object to differentiate between the species. The proposed model localizes green turtles using distinctive features, including a smooth carapace and long flippers. In this case, it becomes evident that the model is sensitive not only to structural but also to surface features of the species. Hawksbill turtles are distinguished by their characteristic shell features, particularly the overlapping scutes and serrated posterior shell margin. These findings indicate that the model effectively learns species-specific biological characteristics. It also suppresses irrelevant background features, leading to improved detection performance. In the case of the Olive Ridley turtles, it can be seen that there is a focus on the shell areas, meaning that there is use of the general body shape and shell patterns that might not be very conspicuous. This characteristic enhances the model's performance under varying lighting conditions and in turbid water.

4.4. Runtime Efficiency

Model efficiency represents the evaluation of a proposed model's suitability in real-world scenarios. Therefore, a detection system that can operate on devices with limited resources is required. This inference experiment is running on a Core i7-12700 @ 4.9 GHz desktop CPU and a Mini PC ZX05 with a 3.4 GHz processor. As shown in Figure 9, YOLOv12-hypernano exhibits higher efficiency in terms of computational cost compared to YOLOv5n, YOLOv8, YOLOv10, and YOLOv11. The proposed model achieves 17.33 FPS on a PC desktop processor and 7.28 FPS on a Mini PC, which is 0.28 FPS higher than YOLOv11. However, the proposed model runs 3 FPS slower when compared to YOLOv5-nano, on a computer desktop processor. In comparison, models integration with recent attention mechanisms, such as Bilinear Mapping and Efficient Modulation, exhibits slower inference performance. Specifically, the YOLOv12h+ESC+Bilinear Mapping achieves

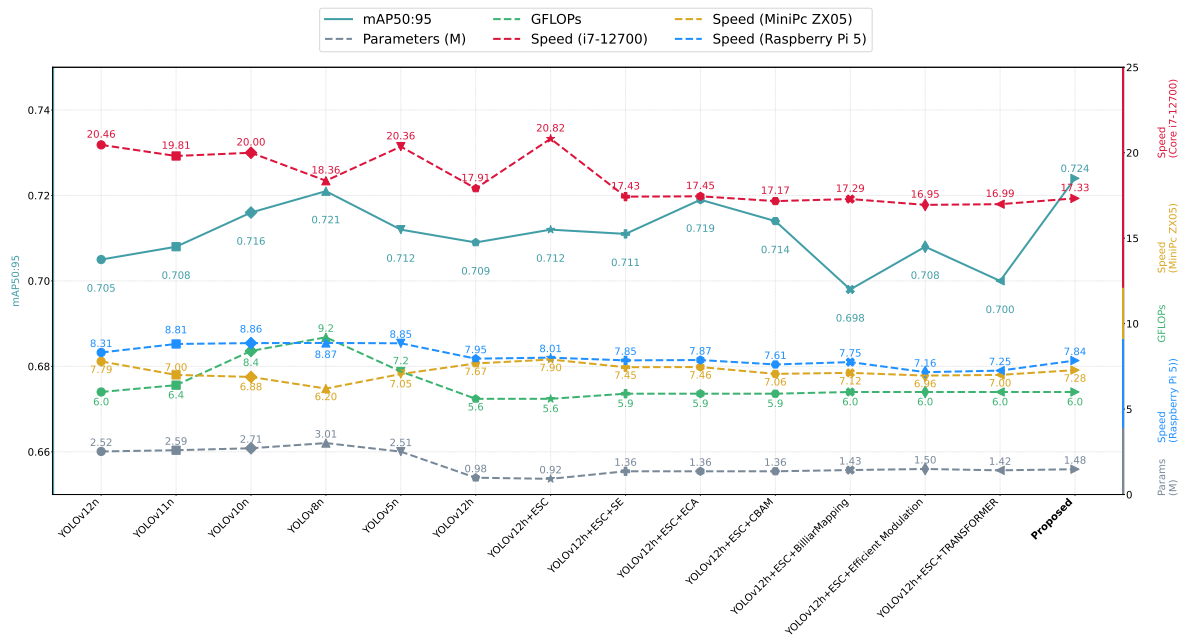


Figure 9. Efficiency comparison of YOLO variants and their modified architectures in terms of computational cost, trainable parameters, and inference speed across two hardware configurations.

17.29 FPS on the desktop CPU and 7.12 FPS on the Mini PC, while the YOLOv12h+ESC+Efficient Modulation records 16.95 FPS and 6.96 FPS on the same devices. These results indicate that, although both modules contribute to effective feature interaction, they introduce a slight computational overhead.

Subsequently, YOLOv10-nano and YOLOv12-nano achieve higher inference speeds on a desktop PC, reaching 20 FPS and 20.46 FPS, respectively. In comparison, YOLOv8-nano and YOLOv11-nano operate at slower speeds than both models. Speed evaluation on a Mini PC shows that YOLOv8-nano and YOLOv10-nano achieve lower inference rates of 6.88 FPS and 6.20 FPS, respectively. Although the proposed model is 3.13 FPS slower than YOLOv12n, it still demonstrates competitive detection performance and remains suitable for deployment on low-cost devices. The lower inference speed occurs because the Efficient Spatial Convolution (ESC) and Dual Spatial Convolution (DSC) modules require large amounts of memory to operate. However, these results still show that the proposed architecture has higher parameter efficiency and improved performance compared to YOLOv12n. In addition, the Lite module incorporates multiple branches and sections, and requires additional memory access, thereby reducing data processing speed. Both proposed modules also utilize depthwise convolution operations which, despite their lightweight computational nature, increase memory access demands. In this work, the model is further evaluated through deployment on a Raspberry Pi 5 device equipped with a 2.4 GHz quad-core ARM Cortex-A76 CPU and 8 GB of RAM, as shown in Figure 9. This compact platform is widely used for sensor and actuator processing in applications such as underwater robotics and Internet of Things (IoT) monitoring systems. Compared to other mini PCs (e.g., ZX05 Mini PC), it offers lower power consumption, making it suitable for battery-powered operation. Performance duration tests indicate that the proposed vision system can operate continuously for up to 201 minutes on a Raspberry Pi 5 powered by a 10,000 mAh power bank (15W, 5V/3A). This implementation leverages floating-point optimization through the OpenVINO framework, reducing memory usage and computational overhead. As a result, the proposed model achieves 7.84 FPS on this device. However, the YOLOv12n-Turbo model [37], which serves as the baseline for the proposed approach, does not incorporate Flash Attention optimization. Consequently, YOLOv12n and its variants exhibit lower inference speed on the Raspberry Pi 5 compared to other lightweight YOLO models. Nevertheless, the proposed sea turtle recognition model achieves higher precision than competing approaches.

5. CONCLUSIONS

This paper presents a sea turtle detector capable of operating in real-time across diverse underwater environments using deep learning techniques. The proposed network is a modified YOLOv12-nano, incorporating several enhancements to improve detection performance under challenging conditions such as extreme illumination, blurring, and distortions that often obscure critical features. To address this problem, the YOLOv12-nano needs to be improved in the extraction process to discriminate the relevant information. The proposed network introduces Efficient Spatial Convolution (ESC) at high-level layers to refine feature representation using the Dual Spatial Convolution (DSC) module. It captures a large area receptive field utilizing depthwise convolution with a 7×7 kernel size, which also maintains the efficiency and creates better spatial relationships. The implementation of the Lite Module utilizes increased feature diversity and selectivity, resulting in improved detection accuracy without significantly increasing model complexity. The evaluation results show that the proposed model successfully achieves mAP95 of 0.724 higher 1.9% than the YOLOv12-nano original. Furthermore, runtime evaluation shows that the proposed model achieves 17.33 FPS on a Core i7-12700 CPU, 7.28 FPS on a ZX05 Mini PC, and 7.84 FPS on a Raspberry Pi 5. Although it is slightly slower than YOLOv12-nano, the proposed model delivers higher precision with fewer parameters, indicating a reasonable trade-off between efficiency and performance.

ACKNOWLEDGMENTS

This research was supported by the Directorate of Research, Technology, and Community Service, Ministry of Education, Culture, Research, and Technology, Indonesia, through the 2025 Young Lecturer Research Scheme under Grant No. 137/C3/DT.05.00/PL/2025, No. 865/LL16/AL.04/2025, and No. 06/PKS-IA/LPPPM/UNIV-T/VI/2025.

REFERENCES

- [1] J. R. Perrault, M. Levin, C. R. Mott, C. M. Boverly, M. J. Bresette, R. M. Chabot, C. R. Gregory, J. R. Guertin, S. E. Hirsch, B. W. Ritchie, S. T. Weege, R. C. Welsh, B. E. Witherington, and A. Page-Karjian, "Insights on Immune Function in Free-Ranging Green Sea Turtles (*Chelonia mydas*) with and without Fibropapillomatosis," *Animals*, vol. 11, no. 3, 2021, doi: 10.3390/ani11030861.
- [2] A. J. Paul, "The Need and Status of Sea Turtle Conservation and Survey of Associated Computer Vision Advances," in *Proc. 2021 IEEE 8th Uttar Pradesh Section Int. Conf. on Electrical, Electronics and Computer Engineering (UPCON)*, 2021, pp. 1–8, doi: 10.1109/UPCON52273.2021.9667626.
- [3] N. Noguchi, H. Nishizawa, T. Shimizu, J. Okuyama, S. Kobayashi, K. Tokuda, H. Tanaka, and S. Kondo, "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.
- [4] C. Rovetto, E. Cruz, E. Flores, I. Nuñez, K. Rodriguez, and E. Cano, "Behavioral data analysis of sea turtles from the Pacific coast of Panama, using biotelemetry," in *Proc. 2023 VI Congreso Internacional en Inteligencia Ambiental, Ingeniería de Software y Salud Electrónica y Móvil (AmITIC)*, 2023, pp. 1–7, doi: 10.1109/AmITIC60194.2023.10366354.
- [5] U. N. Dulhare and M. H. Ali, "Underwater human detection using faster R-CNN with data augmentation," *Materials Today: Proc.*, vol. 80, no. 3, pp. 1940–1945, 2023.
- [6] J. Ding, J. Hu, J. Lin *et al.*, "Lightweight enhanced YOLOv8n underwater object detection network for low light environments," *Scientific Reports*, vol. 14, p. 27922, 2024, doi: 10.1038/s41598-024-27922-7.
- [7] C. Prathima, C. Silpa, A. Charitha, G. Harshitha, C. Sai Charan, and G. R. Sailendra, "Detecting and recognizing marine animals using advanced deep learning models," in *Proc. 2024 Int. Conf. on Expert Clouds and Applications (ICOECA)*, 2024, pp. 950–955, doi: 10.1109/ICOECA62351.2024.00167.
- [8] S. Cai, X. Zhou, W. Cai *et al.*, "Lightweight underwater object detection method based on multi-scale edge information selection," *Scientific Reports*, vol. 15, p. 27681, 2025, doi: 10.1038/s41598-025-13566-3.
- [9] J. Zhang, W. Yongpan, X. Xianchong, L. Yong, L. Lyu, and Q. Wu, "YoloXT: A object detection algorithm for marine benthos," *Ecological Informatics*, vol. 72, p. 101923, 2022, doi: 10.1016/j.ecoinf.2022.101923.
- [10] J. Ma, Y. Zhou, Z. Zhou, Y. Zhang, and L. He, "Toward smart ocean monitoring: Real-time detection of marine litter using YOLOv12 in support of pollution mitigation," *Marine Pollution Bulletin*, vol. 217, p. 118136, 2025.
- [11] F. Zhang, W. Cao, J. Gao, S. Liu, C. Li, K. Song, and H. Wang, "Underwater Object Detection Algorithm Based on an Improved YOLOv8," *Journal of Marine Science and Engineering*, vol. 12, no. 11, p. 1991, 2024.
- [12] H. Zhou, M. Kong, H. Yuan, Y. Pan, X. Wang, R. Chen, W. Lu, R. Wang, and Q. Yang, "Real-time underwater object detection technology for complex underwater environments based on deep learning," *Ecological Informatics*, vol. 82,

- p. 102680, 2024.
- [13] G. Yu, R. Cai, J. Su, M. Hou, and R. Deng, "U-YOLOv7: A network for underwater organism detection," *Ecological Informatics*, vol. 75, p. 102108, 2023.
 - [14] O. M. Lawal, Y. Tan, and C. Liu, "MADNet: Marine Animal Detection Network using the YOLO platform," *PLOS ONE*, vol. 20, no. 5, pp. 1–23, May 2025, doi: 10.1371/journal.pone.0322799.
 - [15] J. Ding, J. Hu, J. Lin *et al.*, "Lightweight enhanced YOLOv8n underwater object detection network for low light environments," *Scientific Reports*, vol. 14, p. 27922, 2024.
 - [16] S. Cai, X. Zhou, W. Cai *et al.*, "Lightweight underwater object detection method based on multi-scale edge information selection," *Scientific Reports*, vol. 15, p. 27681, 2025.
 - [17] Q. Li and H. Shi, "YOLO-GE: An Attention Fusion Enhanced Underwater Object Detection Algorithm," *Journal of Marine Science and Engineering*, vol. 12, no. 10, p. 1885, 2024.
 - [18] Z. Li, B. Zheng, D. Chao, W. Zhu, H. Li, J. Duan, X. Zhang, Z. Zhang, W. Fu, and Y. Zhang, "Underwater-Yolo: Underwater Object Detection Network with Dilated Deformable Convolutions and Dual-Branch Occlusion Attention Mechanism," *Journal of Marine Science and Engineering*, vol. 12, no. 12, p. 2291, 2024.
 - [19] I. R. Oviya, S. A. Rasool, and B. Bhasuran, "Enhanced underwater object detection using deep neural networks," in *Proc. 15th Int. Conf. Comput. Commun. Netw. Technol. (ICCCNT)*, 2024, pp. 1–9.
 - [20] U. N. Dulhare and M. H. Ali, "Underwater human detection using faster R-CNN with data augmentation," *Materials Today: Proceedings*, vol. 80, no. 3, pp. 1940–1945, 2023.
 - [21] M. Liu, Y. Wu, R. Li, and C. Lin, "LFN-YOLO: Precision underwater small object detection via a lightweight reparameterized approach," *Frontiers in Marine Science*, vol. 11, 2025.
 - [22] I. Hipiny, H. Ujir, A. Mujahid, and N. K. Yahya, "Towards Automated Biometric Identification of Sea Turtles (*Chelonia mydas*)," *Journal of ICT Research and Applications*, vol. 12, no. 3, pp. 256–266, Dec. 2018.
 - [23] J. F. Franco-López, M. S. García-Vázquez, and A. Á. Ramírez-Acosta, "Identification of sea turtles with graphical convolutional networks and SLIC superpixel segmentation," in *Proc. SPIE 13136, Optics and Photonics for Information Processing XVIII*, vol. 13136, p. 1313607, Sep. 2024, doi: 10.1117/12.3028445.
 - [24] "Deep Learning for Sea Turtle Classification: A Bibliometric Analysis Using VOSviewer," *Int. J. Inform. Inf. Sys. Comput. Eng.*, vol. 5, no. 1, pp. 88–101, Mar. 2024, doi: 10.34010/injiscom.v5i1.12487.
 - [25] 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.
 - [26] M. D. Putro, *et al.*, "An efficient and effective sea turtle detection using positioning enhancement module," in *Proc. 2024 International Workshop on Intelligent Systems (IWIS)*, 2024, pp. 1–6, doi: 10.1109/IWIS62722.2024.10706029.
 - [27] P. C. Gray, *et al.*, "A convolutional neural network for detecting sea turtles in drone imagery," *Methods in Ecology and Evolution*, vol. 10, no. 3, pp. 345–355, 2019.
 - [28] L. Adam, V. Čermák, K. Papafitsoros, and L. Pícek, "SeaTurtleID2022: A long-span dataset for reliable sea turtle re-identification," *arXiv preprint arXiv:2211.10307*, 2024.
 - [29] J. Zhu, X. Chen, K. He, Y. LeCun, and Z. Liu, "Transformers without Normalization," *arXiv preprint arXiv:2503.10622*, 2025.
 - [30] Z. Zheng, P. Wang, D. Ren, W. Liu, R. Ye, Q. Hu, and W. Zuo, "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.
 - [31] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang, "Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection," *arXiv preprint arXiv:2006.04388*, 2020.
 - [32] A. U. Ruby, P. Theerthagiri, I. J. Jacob, and Y. Vamsidhar, "Binary cross entropy with deep learning technique for image classification," *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 9, no. 4, pp. 5393–5397, 2020, doi: 10.30534/ijatcse/2020/175942020.
 - [33] A. S. Silva, F. H. A. Moraes Neto, P. H. Ferreira and D. B. Costa, "CNN-Based YOLOv12 for Damage Assessment in Residential Roofs," in *IEEE Access*, vol. 13, pp. 193311–193322, 2025, doi: 10.1109/ACCESS.2025.3629630.
 - [34] Z. Peng, Y. Zhang, J. Li, and S. Wang, "Star with Bilinear Mapping," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Nashville, TN, USA, pp. 25292–25302, 2025, doi: 10.1109/CVPR52734.2025.02355.
 - [35] X. Ma, X. Dai, J. Yang, B. Xiao, Y. Chen, Y. Fu, and L. Yuan, "Efficient Modulation for Vision Networks," *arXiv preprint arXiv:2403.19963*, 2024.
 - [36] M. B. Muhammad and M. Yeasin, "Eigen-CAM: Class activation map using principal components," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, pp. 1–7, 2020, doi: 10.1109/IJCNN48605.2020.9206626.
 - [37] Sunsmarterjie, "Releases · sunsmarterjie/yolov12," GitHub, Jul. 2025. <https://github.com/sunsmarterjie/yolov12/releases>.

BIOGRAPHIES OF AUTHORS

Yuliana Mose   Yuliana Mose, S.Kom., M.Si. She completed her bachelor's degree in Informatics Technology in 1998 at the Palapa Institute of Science and Technology, Malang. Then, in 2011, she completed a master's program in Development Resource Management at Sam Ratulangi University. Since 2015, she has served as a permanent lecturer in the Computer Systems study program at Trinita College of Computer Science in Manado, North Sulawesi. She was declared a Professional Lecturer by the Ministry of Education, Culture, Research, and Technology of the Republic of Indonesia. She received the 2nd Rank Award at the Campus Integrity Fest 2023 competition, organized by KPK-RI as a Supervisor in the field of development research. He can be contacted at email: yulimose@gmail.com.



Alex Copernikus Andaria    received his Bachelor of Engineering (B.Eng.) degree in Electrical Engineering from Sam Ratulangi University, Manado, Indonesia, in 2010. In 2022, he earned his Master of Education (M.Pd.) degree in Vocational and Technical Education from Manado State University, Indonesia. He is currently an Assistant Professor in the Bachelor Program of Computer Systems at Trinita University, where he also serves as the Head of the Institute for Research, Community Service, and Quality Assurance. His research interests focus on Electronics and the Internet of Things (IoT), with particular emphasis on the integration of Machine Learning for sensor data processing and the development of adaptive and efficient intelligent systems. He can be contacted via email at andaria.alex@gmail.com.



Hebron Prasetya received his Bachelor of Informatics (S.Kom.) degree from the Bachelor of Informatics Program, Bachelor's 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.



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 (AIVISION) research group. His current research interests include computer vision and deep learning, with a particular focus on robotic vision and perception. He can be contacted at email: dwisnantoputro@unsrat.ac.id.