Jurnal Telematika vol. 20 no. 2 2025
e-ISSN: 2579-3772
doi: 10.61769/telematika.v20i2.799

Comparative Study of Lightweight YOLOv12 Models for Real-Time Underwater Object Detection

Hebron Prasetya[1], Revin R. Balo[2], Tasya Tumbal[3], Alwin M. Sambul[4], Muhamad Dwisnanto Putro[5,*]

[1,2,3,4,5]*Master Program of Informatics, Postgraduate Program, Sam Ratulangi University*
Jl. Kampus Unsrat Bahu, Manado, Sulawesi Utara, Indonesia

[1]hebronprasetya111@student.unsrat.ac.id
[2]revinbalo111@student.unsrat.ac.id
[3]tasyatumbal111@student.unsrat.ac.id
[4]asambul@gmail.com

*Correspondence: [5]dwisnantoputro@unsrat.ac.id

Abstract*— Deep learning methods in computer vision play a crucial role in object localization using camera-based sensors, with Convolutional Neural Networks serving as the dominant approach for object detection. However, many existing models incur high computational costs due to deep architectures and complex operations, limiting their use for real-time deployment on low-cost, resource-constrained devices. The YOLOv12 architecture offers lightweight variants to improve computational efficiency. This study evaluates the trade-off between efficiency and detection performance by comparing model variants using the number of parameters, floating-point operations, and inference speed, while detection accuracy is measured using mean average precision. The results assess the suitability of lightweight models for real-time deployment in resource-constrained environments such as underwater monitoring and conservation. Experimental results on the Real-World Underwater Object Detection dataset demonstrate that YOLOv12-nano achieves 5.7% lower accuracy compared to YOLOv12-medium but requires only 2.57 million parameters and 6.5 GFLOPs, significantly less than YOLOv12-medium with 20.1 million parameters and 67.8 GFLOPs. Moreover, YOLOv12-small requires 9.26 million parameters and 21.5 GFLOPs, positioning it between nano and medium in terms of complexity while still maintaining competitive accuracy. In the inference process, YOLOv12-nano achieves 16.48 FPS on a 12th Gen Intel(R) Core (TM) i5-12450HX CPU. In comparison, YOLOv12-small runs at 6.28 FPS, while YOLOv12-medium runs at 2.36 FPS. These results indicate that YOLOv12-nano is the most suitable variant for real-time deployment on CPU-based platforms.

***Keywords*— underwater, object detection, convolutional neural network, lightweight YOLOv12, efficient model**

***Abstract*— Metode *deep learning* dalam *computer vision* berperan penting dalam pelokalan objek menggunakan sensor berbasis kamera dengan *Convolutional Neural Networks* sebagai pendekatan utama dalam deteksi objek. Namun, banyak model yang ada memiliki biaya komputasi yang tinggi akibat arsitektur yang dalam dan operasi yang kompleks sehingga membatasi penerapannya untuk kebutuhan waktu nyata pada perangkat berbiaya rendah dan dengan sumber daya terbatas. Arsitektur YOLOv12 menawarkan beberapa varian ringan yang dirancang untuk meningkatkan efisiensi komputasi. Penelitian ini mengevaluasi keseimbangan antara efisiensi dan kinerja deteksi dengan membandingkan berbagai varian model berdasarkan jumlah parameter, operasi *floating-point*, dan kecepatan inferensi, serta mengukur akurasi menggunakan *mean average precision*. Hasil evaluasi ini digunakan untuk menilai kesesuaian model yang ringan dalam penerapan waktu nyata pada lingkungan dengan sumber daya terbatas, seperti pemantauan dan konservasi bawah air. Hasil eksperimen pada *dataset real-world underwater object detection* menunjukkan bahwa YOLOv12-nano memiliki akurasi 5,7% lebih rendah dibandingkan YOLOv12-medium, namun hanya membutuhkan 2,57 juta parameter dan 6,5 GFLOPs, jauh lebih kecil dibandingkan YOLOv12-medium yang memiliki 20,1 juta parameter dan 67,8 GFLOPs. Selain itu, YOLOv12-small membutuhkan 9,26 juta**

parameter dan 21,5 GFLOPs sehingga berada di antara varian nano dan medium dari sisi kompleksitas, dengan akurasi yang tetap kompetitif. Pada proses inferensi, YOLOv12-nano mencapai kecepatan 16,48 FPS pada CPU Intel(R) Core (TM) i5-12450HX generasi ke-12. Sebagai perbandingan, YOLOv12-small berjalan pada 6,28 FPS, sedangkan YOLOv12-medium mencapai 2,36 FPS. Hasil ini menunjukkan bahwa YOLOv12-nano merupakan varian yang paling sesuai untuk penerapan waktu nyata pada *platform* berbasis CPU.

Kata kunci— bawah laut, deteksi objek, convolutional neural network, YOLOv12 ringan, model efisien

I. INTRODUCTION

Traditional conservation methods involve high operational costs and pose risks to human safety. In addition, these challenges can be overcome through an automated system capable of detecting and locating objects underwater [1], [2]. One of the widely known methods for localization tasks is the convolutional neural network (CNN) [3]. However, achieving high detection performance often requires CNN methods to employ deeper architectures with many layers and convolutional filters, which significantly increases computational cost. This complexity limits the feasibility of such models for real-time deployment on low-cost and resource-constrained platforms. A comparative experimental study is essential to determine which model variant offers efficient deployment across different devices while maintaining competitive performance. Variations in lighting due to depth and water turbidity cause uneven illumination, while suspended particles scatter light, leading to blurring, color distortion, and reduced contrast [4]. Therefore, it is essential to evaluate deep learning architectures that can effectively handle complex underwater visual characteristics while maintaining efficient, reliable, real-time performance.

A previous study proposes a lightweight underwater object detection model based on a modified YOLOv8-nano architecture, called RDL-YOLO [5]. The model uses the same dataset for training and performance evaluation. However, the comparison focuses solely on model size and computational complexity, reporting only limited metrics, such as the number of parameters and floating-point operations. RDL-YOLO contains 2.43 million parameters and requires 6.9 GFLOPs, which are higher than those of YOLOv12-nano. Furthermore, the lack of an inference speed evaluation limits the assessment of its suitability for real-time deployment. Another study [6] introduces EAST-YOLO, which adopts the YOLO11-nano architecture. The model contains 2.6 million parameters and requires 6.5 GFLOPs, which are comparable to YOLOv12-nano. The study evaluates efficiency and detection performance using metrics such as GFLOPs and mean average precision. However, the analysis is limited to the nano variant, preventing a comprehensive comparison across different model scales, such as small and medium variants. Subsequently, study [7] proposed a lightweight detection model named YOLO-Fast. However, the comparison used only the small-scale variant, YOLOv8s, as the baseline. This limitation leaves a research gap in evaluating nano-scale variants for efficiency and real-time deployment.

CNN architectures rely on large numbers of parameters and high computational complexity, which result in high computational costs [8]. The work [9] proposes a two-stage underwater object detector based on a region-based convolutional neural network (R-CNN) and a swin transformer to improve detection performance. However, it increases computational complexity due to the use of self-attention, achieving only 12.8 FPS, which is slower than YOLO architectures. Furthermore, the work in [10] introduces multiple convolutional blocks to enhance the performance. It employs deformable large kernel attention (D-LKA), which relies on large kernel operations, and incorporates separate and enhancement attention modules (SEAM), both of which increase computational demands.

In addition, underwater imagery suffers from light scattering, color distortion, and low visibility, which degrade image quality [11], [12], [13]. Therefore, robust models with sufficient convolutional depth are required to extract meaningful features from complex data. Balancing lightweight design with strong feature representation is crucial to achieve both accuracy and real-time performance in underwater detection tasks [14]. YOLOv12 is a CNN designed for real-time object detection [15]. It offers nano, small, and medium model sizes. The nano model is the fastest and most efficient for real-time use. The small and medium models are more accurate but need more computation. Deeper models produce better features but slow down processing, which makes them less suitable for real-time tasks. This work addresses the limited investigation of how different YOLO model scales influence detection accuracy, computational cost, and inference speed [5]. Existing studies focus only on nano-scale models [6]. Other works rely on traditional architectures [9], resulting in an incomplete understanding of scalability trade-offs. Furthermore, this study introduces a new underwater object detection framework based on the YOLOv12 architecture, trained and evaluated on the real-world underwater object detection dataset. The comparative analysis aims to identify which variant achieves the optimal balance between detection performance, inference speed, and computational efficiency across various devices. This study also compared the small and medium variants of YOLOv12 to evaluate trade-offs in computational cost and detection performance. The study seeks to determine the most suitable configuration for real-time applications in resource-constrained environments, such as edge-based marine monitoring and mobile conservation systems.

II. Methodology

A. *YOLOv12*

The You Only Look Once (YOLO) series has become one of the most prominent frameworks in real-time object detection due to its capability to achieve a strong trade-off between accuracy and computational efficiency. As shown in Figure 1, the YOLOv12 network comprises several area attentions with C2f (A2C2f) and cross stage partial with C3k (C3K2) modules to optimize feature extraction, enhance multi-scale feature representation, and improve detection precision through a single-stage detection approach. Unlike traditional two-stage methods that separate region proposal and classification steps, YOLOv12 integrates these steps into a unified pipeline, resulting in faster, more streamlined inference. This design allows the model to operate effectively in real-time applications and on devices with limited computational resources. Additionally, YOLOv12 incorporates refined backbone and neck structures to strengthen feature fusion and localization accuracy while preserving lightweight computation.

YOLOv12 includes five variants: nano, small, medium, large, and extra-large, each balancing accuracy and efficiency for different applications. This study focuses on lightweight nano-, small-, and medium-sized models optimized for real-time performance. The nano version with 2.6 million parameters and 6.7 GFLOPs offers high speed on limited hardware, the small version with 9.3 million parameters and 21.7 GFLOPs provides higher precision, and the medium version with 20.2 million parameters and 68.1 GFLOPs achieves a balanced trade-off between accuracy and efficiency.

B. *Backbone*

The backbone in YOLOv12 serves as the main feature extractor, processing input images to learn useful visual patterns. It uses the A2C2f and C3K2 modules to improve feature quality and efficiency. With a pyramid structure, the backbone can capture features at multiple scales, enabling accurate detection of both small and large objects with minimal computational cost.

1) *A2C2f*: The A2C2f module in YOLOv12 improves feature extraction by combining area attention with efficient convolution. As shown in Figure 2, it starts with a 1×1 convolution to reduce the channel size and then splits into two paths. One path acts as a shortcut, while the other applies either area attention or the C3K block. Area Attention captures global context by modelling relationships between feature regions. The C3K block is a modified C3 structure that utilizes flexible kernel sizes to better capture spatial

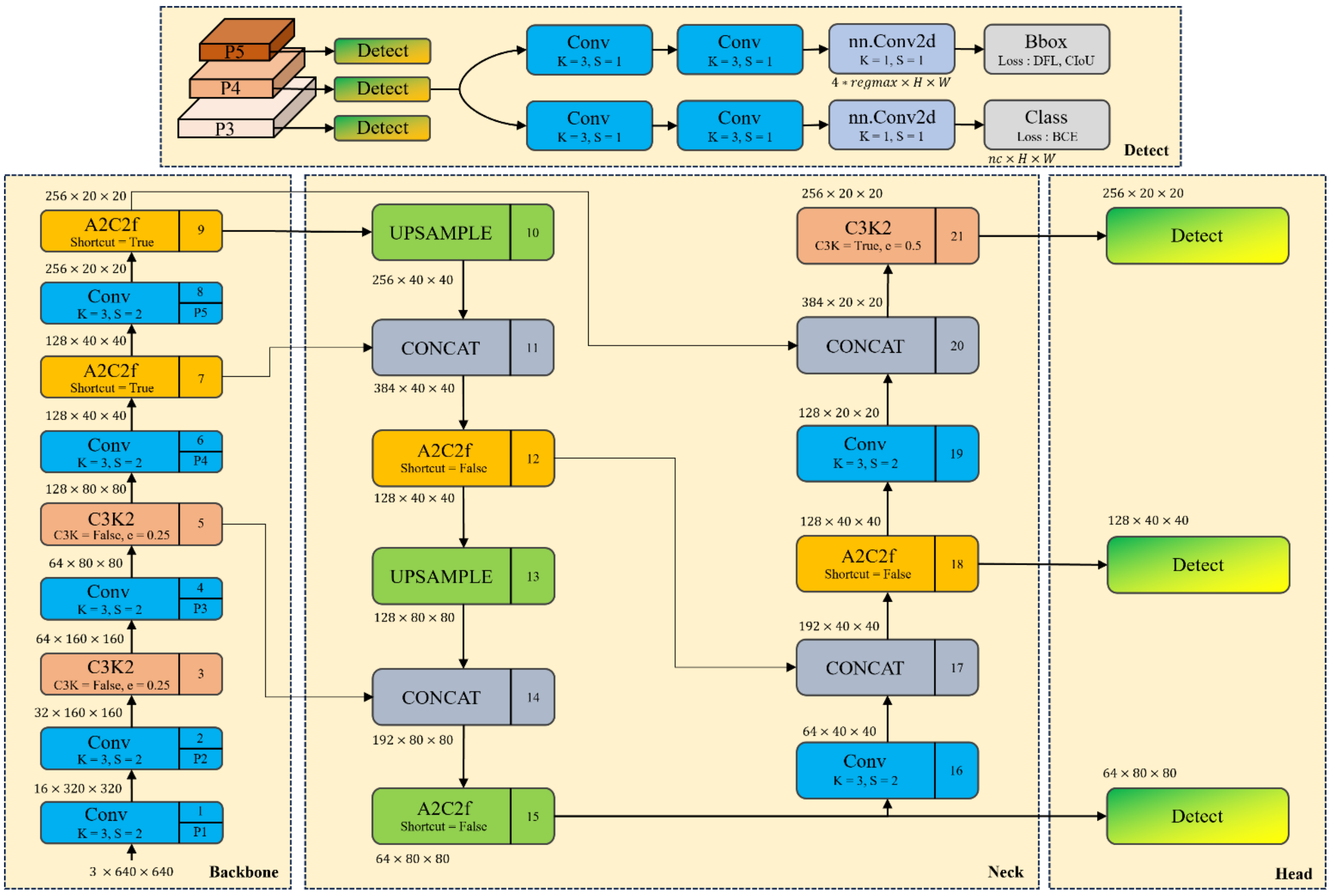


Figure 1 The YOLOv12-nano architecture consists of a backbone (A2C2f, C3K2) for feature extraction, a neck for feature fusion, and a detection head predicting objects on three layers (P3, P4, P5) [15]

information, as shown in Figure 3. The outputs are then processed by a feed-forward layer with two 1×1 convolutions to refine the features.

2) *C3k2*: To address underwater challenges such as blur and low visibility, effective feature extraction is required. As shown in Figure 4, the C3K2 module acts as an efficient feature extractor. It is a lightweight variant of the CSP bottleneck, similar to the C2f structure but using the C3K design. The module starts with a 1×1 convolution to adjust the number of channels. Half of the channels are used for feature extraction, while the remaining channels are preserved as identity connections to improve efficiency. C3K2 supports two modes: C3K, with three convolutions and flexible kernel sizes, and C2f, with two 3×3 convolutions. The outputs are then concatenated and processed by another 1×1 convolution to enhance channel interaction. This compact structure provides strong feature representations for underwater detection.

C. *Neck*

The neck serves as the bridge between the backbone and detection head, merging multi-scale feature maps to produce more discriminative representations. YOLOv12 employs a path aggregation network (PAN) structure to enable both top-down and bottom-up feature fusion, enhancing context awareness across layers [16]. Convolution and C3K2 modules are used to improve feature integration while maintaining computational efficiency.

D. Head

YOLOv12 uses an anchor-free detection head derived from YOLOv8, separating objectness, classification, and bounding-box regression for enhanced accuracy and inference stability. The head consists of two parallel branches with 3×3 and 1×1 convolutions, producing predictions at three scales (80×80, 40×40, and 20×20) to detect small, medium, and large objects. The optimization process uses Distribution Focal Loss (DFL) and Complete Intersection over Union (CIoU) loss for bounding-box regression and Binary Cross-Entropy (BCE) for classification [17], [18], [19].

$$Total_{loss} = \lambda_{CIoU} L_{CIoU} + \lambda_{DFL} L_{DFL} + \lambda_{Cls} L_{Cls}. \tag{1}$$

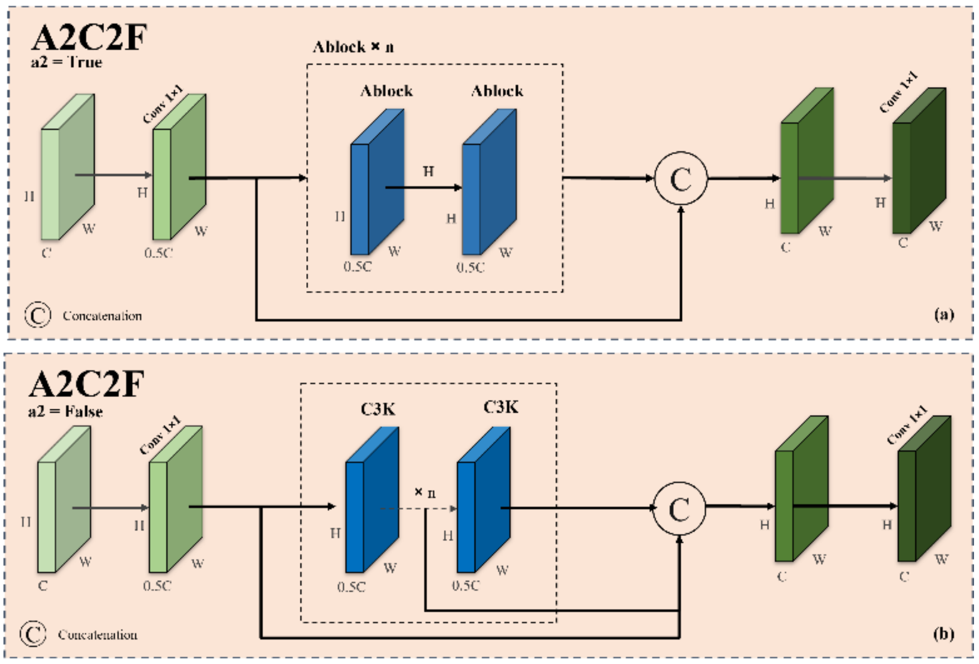


Figure 2 The A2C2f module uses area attention for global feature relations (True) and C3K block for adaptive spatial extraction (False) [15].

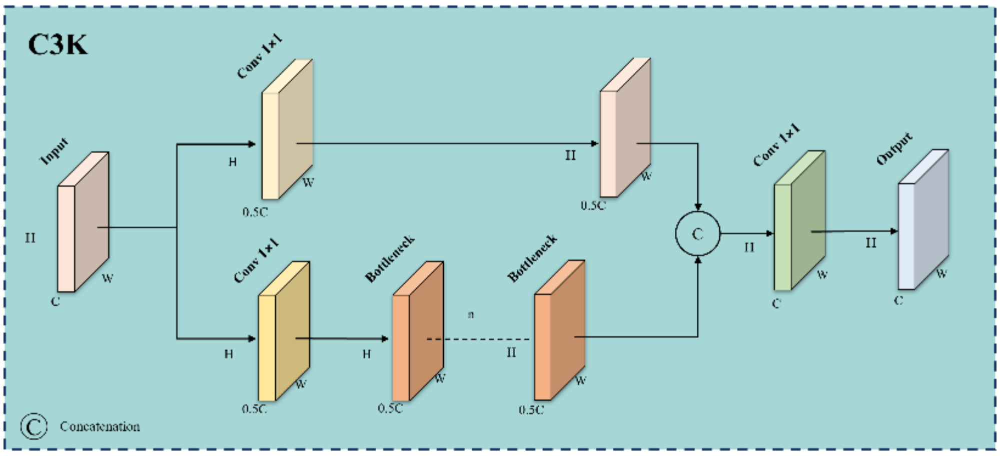


Figure 3 The C3K module structure in YOLOv12, which works together with thebBottleneck module to extract feature information efficiently [15].

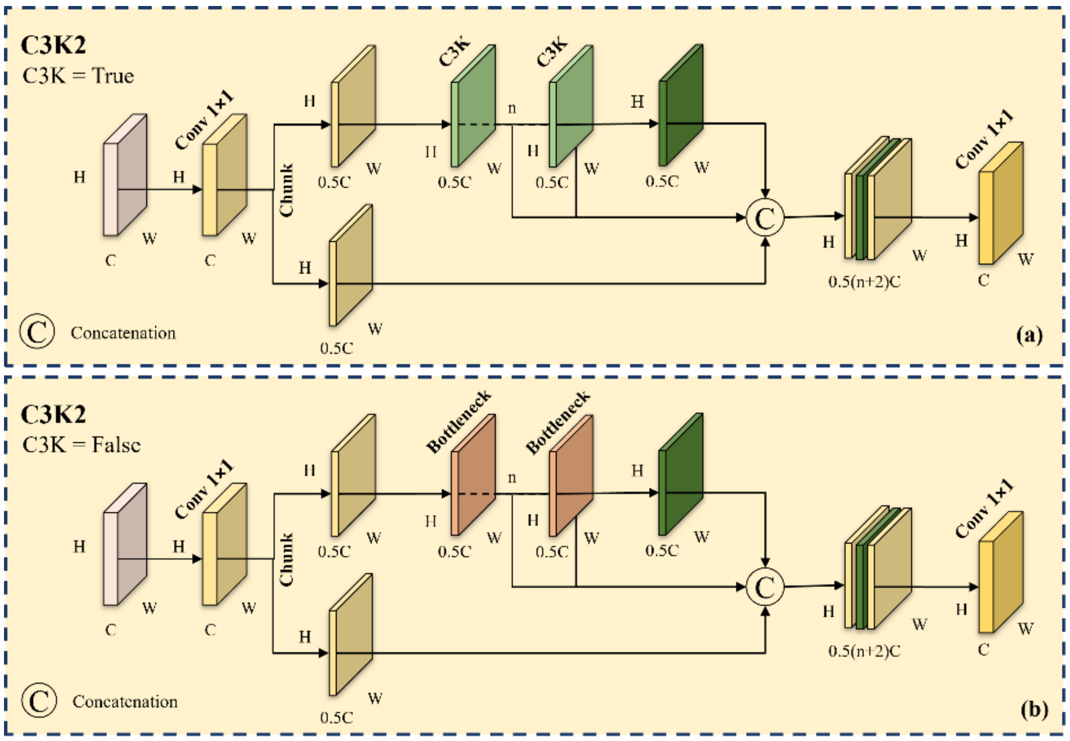


Figure 4 The C3K2 module uses C3K blocks for multi-scale feature aggregation (True) and Bottleneck blocks for efficient local extraction (False) [15].

The total loss comprises of three components. The CIoU loss L_{CIoU} measures the discrepancy between the predicted bounding box and the ground-truth box by considering overlap, distance, and aspect ratio. The distribution loss L_{DFL} applies Distribution Focal Loss to provide smooth supervision over bounding box regression. In addition, the classification loss L_{Cls} employs Binary Cross-Entropy (BCE) to penalize incorrect class predictions.

E. *Comparison Methods*

In this work, efficiency is measured by the number of parameters and giga floating point operations (GFLOPs) to represent computational complexity. Mean average precision (mAP) quantifies the accuracy of localization and classification. Inference speed uses frames per second (fps). These metrics are widely used in computer vision to analyze the trade-off between accuracy and efficiency. The number of parameters depends on the kernel size, the number of input channels, and the number of output channels. This relationship is expressed as follows:

$$Parameters = (K_h \times K_w \times C_{in} + 1) \times C_{out}. \tag{2}$$

The number of parameters is calculated based on the kernel size (representing the kernel height and width), the number of input channels, and a bias term. The total parameter count is obtained by multiplying these values by the number of output channels. In addition, GFLOPs is used to represent computational cost by estimating the number of floating-point operations. This process is expressed as follows:

$$FLOPs = h \times w \times C_{out} \times (k \times k \times C_{in}). \tag{3}$$

The computational cost is calculated by multiplying the total number of output feature map elements $(h \times w \times C_{out})$ by the total number of kernel weights $((k \times k \times C_{in})$. This metric represents the number of floating-point operations required to process the feature maps and is commonly used to estimate computational complexity [20]. In addition, mAP is used to evaluate detection performance. This metric is widely adopted in object detection with thresholds of 50% and 50%-95%, where higher values indicate more precise predictions. This process is defined as follows:

$$mAP = \frac{1}{k} \sum_{i}^{k} AP_i \,. \tag{4}$$

Mean average precision evaluates detection performance by computing predictions using intersection over union (IoU) thresholds of 50% and 50%-95%. The average precision for each class is first calculated, and the final mAP value is obtained by averaging across all classes. This metric enables accurate evaluation of positive sample predictions and overall detection quality.

F. Dataset

In this study, the real-world underwater object detection (RUOD) [21] dataset is used as the benchmark for model evaluation, as shown in Figure 5. RUOD is a large-scale dataset designed to overcome the limitations of previous underwater datasets, which have limited object types and scene diversity. It contains 14,000 high-resolution images and 74,903 annotated objects across 10 aquatic categories, including fish, diver, starfish, corals, turtle, echinus, holothurian, scallop, cuttlefish, and jellyfish. Each image includes bounding box annotations and class labels. The dataset comprises 9,800 training images and 4,200 test images, covering diverse underwater environments collected from public sources. RUOD features a range of real-world challenges, including blur effects, color casts, light interference, and complex marine conditions, making it a comprehensive and reliable benchmark for evaluating underwater object detection performance.

G. Implementations Setup

As shown in Table I, training is performed on the Kaggle platform using an NVIDIA P100 GPU. The model is trained for 150 epochs with a batch size of 16, and the input resolution is set to 640×640 pixels to retain spatial detail. Stochastic gradient descent with a learning rate of 0.01 is used to ensure stable parameter updates. This configuration provides sufficient computational capacity for effective learning while helping prevent overfitting. For inference, Table I shows the model runs on Ubuntu 21 with an Intel i5-12450HX CPU using PyTorch 2.0.1, with input images of 640 × 640 pixels as in training. This setup enables a realistic evaluation of computational efficiency and inference speed on mid-range, GPU-free devices.

III. RESULTS AND DISCUSSIONS

A. Evaluation on Datasets

To evaluate model performance, several key metrics are employed. The number of trainable parameters determines the model's size and memory requirements. Giga floating-point operations (GFLOPs) indicate the computational complexity of a single forward pass and reflect the overall computation cost. Detection accuracy is reported using mAP50 and mAP50:95. In addition, frames per second (fps) represent the inference speed and indicate how many images the model processes per second during inference. These

Figure 5 Sample images from the RUOD dataset showing diverse underwater scenes with various marine species, divers, and environmental conditions such as haze, color distortion, and light interference.

metrics provide a comprehensive analysis of both detection performance and computational efficiency. As shown in Table I, YOLO12n and YOLO11n achieve the highest detection accuracy. YOLO12n obtains an mAP50 of 0.833 and mAP50:95 of 0.590, while YOLO11n slightly improves performance with 0.834 and 0.591, respectively. Despite having similar parameter sizes (approximately 2.6M), YOLO11n demonstrates marginally better accuracy. This indicates that the structural refinements in YOLO11n enhance feature extraction without increasing computational complexity. In contrast, YOLO-Fast [7] has the smallest parameter count (2.34M) and the lowest computational cost (6.5 GFLOPs), but its mAP50:95 score of 0.546 is significantly lower.

The result indicates that although YOLO-Fast is suitable for highly resource-constrained environments, it compromises detection accuracy due to its limited representational capacity. Models such as YOLOv10n [22], YOLOv8n, and YOLOv6n provide moderate performance, with mAP50 scores ranging from 0.812 to 0.830 and parameter sizes between 2.7M and 4.2M, offering a balanced trade-off between accuracy and efficiency. However, they still do not surpass the performance of newer architectures like YOLO11n and YOLO12n. On the higher end, YOLO12s and YOLO12m achieve the strongest results, with YOLO12s reaching an mAP50 of 0.860 and YOLO12m achieving the highest accuracy of 0.871, though these gains come at substantially higher computational cost due to their larger parameter sizes.

Notably, YOLOv12n provides an advantageous balance between performance and efficiency. With approximately 2.57M parameters and 6.5 GFLOPs, it achieves competitive detection accuracy while maintaining a lightweight structure, making it well-suited for real-time applications and deployment on devices with limited hardware resources. As illustrated in Figure 6, both YOLOv12-nano and YOLOv12-small demonstrate effective detection performance across various underwater environments containing fish, holothurians, echinus, starfish, and divers. However, noticeable differences emerge in detection precision and robustness.

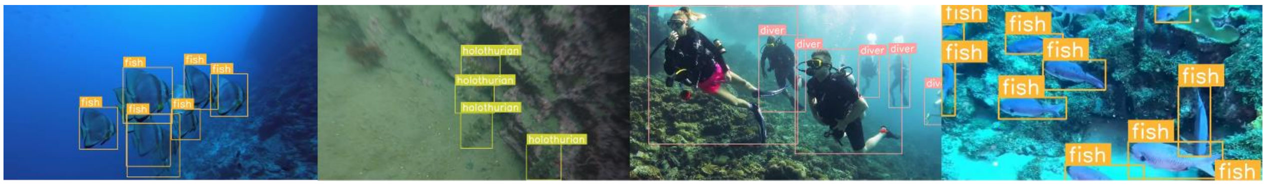
Based on Figure 6, the YOLOv12-nano model in (a) is able to detect the main objects but generates fewer bounding boxes and occasionally misses small or partially occluded targets. This behaviour is expected, as nano-scale models are optimized for lightweight computation rather than extensive feature representation. In comparison, the YOLOv12-small model in (b) provides more complete detection coverage. Selecting the most efficient YOLOv12 variant prioritizes inference speed and computational cost while maintaining competitive detection accuracy. YOLOv12-nano uses 2.5 million parameters and 6.5 GFLOPs, achieving an inference speed of 16.48 FPS. In comparison, YOLOv12-small increases the parameter count by 268% to 9.2 million and the computational cost by 231% to 21.5 GFLOPs, resulting in a 61.89% reduction in inference speed to 6.28 FPS. Although YOLOv12-small achieves a higher mAP, its higher computational demand limits its suitability for low-CPU and edge devices. As shown in Figure 6, YOLOv12-nano provides competitive detection performance compared to the YOLOv12-small variant. Therefore, inference speed and computational efficiency are the primary criteria in selecting the YOLOv12-nano variant.

B. *Runtime Efficiency*

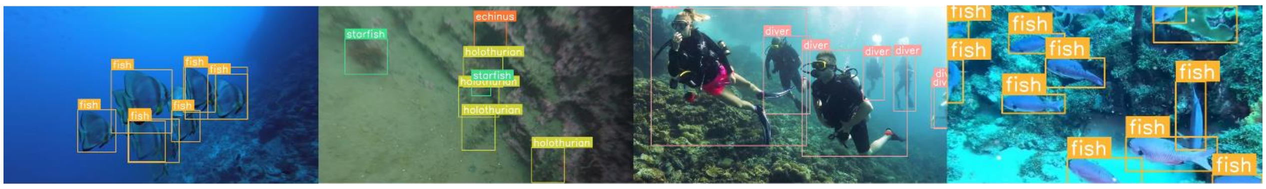
To evaluate runtime efficiency, inference performance was analyzed across several YOLO variants under a CPU-based setup. As shown in Table I, lower parameter counts and GFLOPs do not always guarantee higher FPS, indicating that model speed is influenced not only by computational complexity but also by algorithmic optimization and memory access efficiency. For example, YOLOv10n and YOLO11n achieve the highest FPS values (18.37 and 18.59, respectively) despite having similar or slightly higher parameter counts compared to lighter models like YOLOv12n and YOLO-FAST. This demonstrates that architectural design and efficient memory utilization play a more critical role in determining real-time performance than raw model size alone.

TABLE I
COMPARISON OF YOLO VARIANTS USING MODEL SIZE, COMPUTATIONAL COST, ACCURACY, AND SPEED

Models	mAP50	mAP50:95	Parameters	GFLOPs	FPS
YOLO-FAST	0.807	0.546	2,340,366	6.5	15.32
YOLOv10n	0.825	0.581	2,710,940	8.4	18.37
YOLOv8n	0.83	0.583	3,012,798	8.2	15.96
YOLOv6n	0.812	0.568	4,239,134	11.9	13.84
YOLOv12s	0.86	0.628	9,257,006	21.5	6.28
YOLOv12m	0.871	0.647	20,145,198	67.8	2.36
YOLOv12n-Turbo	0.828	0.585	2,521,614	6.0	16.49
YOLO11n	0.834	0.591	2,591,790	6.5	18.59
YOLOv12n	0.833	0.59	2,569,998	6.5	16.48



(a)



(b)

Figure 6 Qualitative detection results on representative samples from the RUOD dataset. (a) Detection generated by the YOLOv12-nano model. (b) Detection generated by the YOLOv12-small model.

YOLOv12n and YOLO11n both maintain competitive speeds of 16.48 FPS and 18.59 FPS, respectively, while keeping computational cost low at 6.5 GFLOPs, making them suitable for resource-constrained deployment. In contrast, larger models such as YOLOv12s and YOLOv12m show significantly reduced FPS with 6.28 and 2.36, respectively. These results highlight that YOLOv12n achieves a favorable balance between inference speed and detection performance, making it well-suited for real-time applications on mid-range hardware.

IV. CONCLUSIONS

This study evaluates lightweight YOLOv12 variants, aiming to achieve an optimal balance between detection accuracy and computational efficiency for real-time object detection. YOLOv12-nano achieves 16.48 FPS on CPU inference while maintaining competitive detection accuracy with an mAP50:95 of 0.590, utilizing only 2.57 million parameters and 6.5 GFLOPs. These results highlight its suitability for edge-based or low-power systems, such as underwater monitoring platforms, where real-time processing and energy efficiency are critical. Meanwhile, the small and medium variants offer higher accuracy but require greater computational resources, making them more appropriate for GPU-based environments. This efficiency result highlights the potential to further improve detection performance by integrating enhancement modules.

ACKNOWLEDGEMENT

The authors sincerely thank the AIVISION research team for their valuable guidance and expertise in computer vision and deep learning, as well as for providing the computational resources that supported the experiments and refinement of this manuscript.

REFERENCES

[1] A. Apprill *et al.*, “Toward a New Era of Coral Reef Monitoring,” *Environ. Sci. Technol.*, vol. 57, no. 13, pp. 5117–5124, Apr. 2023, doi: 10.1021/acs.est.2c05369.

[2] F. Wu, Z. Cai, S. Fan, R. Song, L. Wang, and W. Cai, “Fish Target Detection in Underwater Blurred Scenes Based on Improved YOLOv5,” *IEEE Access*, vol. 11, pp. 122911–122925, 2023, doi: 10.1109/ACCESS.2023.3328940.

[3] 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.

[4] P. Pachaiyappan, G. Chidambaram, A. Jahid, and M. H. Alsharif, “Enhancing Underwater Object Detection and Classification Using Advanced Imaging Techniques: A Novel Approach with Diffusion Models,” *Sustainability*, vol. 16, no. 17, 2024, doi: 10.3390/su16177488.

[5] D. Song and H. Huo, “Lightweight Underwater Target Detection Algorithm Based on YOLOv8n,” *Electronics (Basel).*, vol. 14, no. 9, 2025, doi: 10.3390/electronics14091810.

[6] Y. Xu and X. Xiao, “Exploring the Depth From EAST: Efficient Aggregated State-Space Tanh-Tuned Model for Underwater Object Detection,” *IEEE Signal Process. Lett.*, vol. 32, pp. 3809–3813, 2025, doi: 10.1109/LSP.2025.3606841.

[7] Z. Song, X. Zhang, and P. Tan, “YOLO-Fast: a lightweight object detection model for edge devices,” *Journal of Supercomputing*, vol. 81, no. 5, Apr. 2025, doi: 10.1007/s11227-025-07172-3.

[8] J. Lei, H. Wang, Z. Lei, J. Li, and S. Rong, “CNN–Transformer Hybrid Architecture for Underwater Sonar Image Segmentation,” *Remote Sens. (Basel).*, vol. 17, no. 4, 2025, doi: 10.3390/rs17040707.

[9] J. Liu, S. Liu, S. Xu, and C. Zhou, “Two-Stage Underwater Object Detection Network Using Swin Transformer,” *IEEE Access*, vol. 10, pp. 117235–117247, 2022, doi: 10.1109/ACCESS.2022.3219592.

[10] L. Guo, X. Liu, D. Ye, X. He, J. Xia, and W. Song, “Underwater object detection algorithm integrating image enhancement and deformable convolution,” *Ecol. Inform.*, vol. 89, p. 103185, 2025, doi: https://doi.org/10.1016/j.ecoinf.2025.103185.

[11] X. Qin, C. Yu, B. Liu, and Z. Zhang, “YOLO8-FASG: A High-Accuracy Fish Identification Method for Underwater Robotic System,” *IEEE Access*, vol. 12, pp. 73354–73362, 2024, doi: 10.1109/ACCESS.2024.3404867.

[12] W. Yi, J. Yang, and L. Yan, "Research on Underwater Small Target Detection Technology Based on Single-Stage USSTD-YOLOv8n," *IEEE Access*, vol. 12, pp. 69633–69641, 2024, doi: 10.1109/ACCESS.2024.3400962.

[13] 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.

[14] 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.

[15] Y. Tian, Q. Ye, and D. Doermann, "YOLOv12: Attention-Centric Real-Time Object Detectors," Feb. 2025, [Online]. Available: http://arxiv.org/abs/2502.12524

[16] 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*, 2018, pp. 8759–8768. doi: 10.1109/CVPR.2018.00913.

[17] X. Li *et al.*, "Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection," Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.04388

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

[19]. Usha Ruby Dr.A, "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, Aug. 2020, doi: 10.30534/ijatcse/2020/175942020.

[20] N. Shahadat and A. S. Maida, "Analyzing Parameter-Efficient Convolutional Neural Network Architectures for Visual Classification," *Sensors*, vol. 25, no. 24, Dec. 2025, doi: 10.3390/s25247663.

[21] C. Fu *et al.*, "Rethinking general underwater object detection: Datasets, challenges, and solutions," *Neurocomputing*, vol. 517, pp. 243–256, 2023, doi: https://doi.org/10.1016/j.neucom.2022.10.039.

[22] A. Wang *et al.*, "YOLOv10: Real-Time End-to-End Object Detection," Oct. 2024, [Online]. Available: http://arxiv.org/abs/2405.14458

Hebron Prasetya. Earned his Bachelor's degree in Informatics from the Informatics Program at Sam Ratulangi University, Manado, Indonesia, in 2025. He is currently enrolled in the Master's program in Informatics at the same university and is actively involved as a member of the Algorithmic Intelligence for Vision (AIVISION) research group. His research focuses on computer vision and deep learning.

Revin Rehuel Balo. Began his undergraduate studies in 2020 and earned his Bachelor's degree in Electrical Engineering from the Department of Electrical Engineering, Sam Ratulangi University, Manado, Indonesia, in 2024. He is currently pursuing his Master's degree in the Master Program of Informatics, Postgraduate Program, Sam Ratulangi University, Manado, Indonesia.

Tasya Tumbal. Obtained her Bachelor's degree in Informatics from Sam Ratulangi University, Manado, Indonesia, in 2024. She is currently pursuing her Master's degree in the Master Program of Informatics, Postgraduate Program, Sam Ratulangi University, Manado, Indonesia.

Alwin M. Sambul. Received his Bachelor's degree in Engineering from Universitas Sam Ratulangi, Manado, Indonesia, in 2003. He obtained his Master of Engineering degree from Kumamoto University, Japan, in 2011, and completed his Doctor of Philosophy degree at Kumamoto University, Japan, in 2015. His research interests include Computer Science, Biomedical Engineering, Biomedical Informatics, and Educational Technology

Muhamad Dwisnanto Putro. Obtained his Bachelor of Engineering degree in Electrical Engineering from Sam Ratulangi University, Manado, Indonesia, in 2010. He completed his Master of Engineering (M.Eng.) degree at the Department of Electrical Engineering, Gadjah Mada University, Yogyakarta, Indonesia, in 2012, and earned his Ph.D. in Electrical, Electronic, and Computer Engineering from the University of Ulsan, South Korea, in 2022. He currently holds the position of Associate Professor in the Master Program of Informatics at Sam Ratulangi University and leads the Algorithmic Intelligence for Vision (AI-VISION) research group. His research activities focus on intelligent vision and deep learning, particularly in the areas of robotic vision and perception.