Optimal size reduction methodology for YOLO-based underwater object detectors based on knowledge distillation

Victor Sineglazov[1, 2, *, †], Mykhailo Savchenko [1, †], Michael Z. Zgurovsky [1]

[1] National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", 37, Prospect Beresteiskyi, Kyiv, 03056, Ukraine

[2] National Aviation University, 1, Prospect Liubomyra Huzara, Kyiv, 03058, Ukraine

Abstract

The paper introduces a novel methodology for reducing the size of YOLO-based underwater object detectors to deploy it on edge hardware. Feature extraction layer light-weighting technique is used to compress the model with minimal impact on performance. Two new object detection network topologies are created, suitable to be used as a student network in knowledge distillation tasks. Knowledge distillation algorithm with temperature decay strategy is developed to mitigate the performance loss caused by model compression without inflating the parameter count. Object detection models, based on the proposed methodology, are tested on Underwater Target Detection Algorithm Competition 2020 dataset, providing higher accuracy and offering faster runtime than the existing solutions.

Keywords

underwater object detection, autonomous underwater vehicles, neural networks, deep learning, knowledge distillation

1. Introduction

Due to rapid development of deep learning [1], optimization algorithms [2] and complex neural network topologies [3, 4, 5], developing intellectual systems for autonomous underwater vehicles (AUVs) is gaining more and more attention. These applications include biodiversity exploration, pollution monitoring, demining and surveillance operations, rescue missions, and other critical tasks in underwater environments.

To operate in real-time mode, the artificial intelligence model is usually deployed on a separate device, typically a single board computer, which is then installed on the AUV. This approach allows for greater modularity in system design and easier upgrades. However, this method has its own set of challenges and limitations. The edge devices chosen for this purpose are selected primarily for their economic and power efficiency, which is crucial for underwater operations. In such cases, performance is frequently traded off for extended battery runtime and lower overall system costs, potentially limiting the capabilities of installed software.

Another major problem is that detecting targets in underwater environments is significantly more challenging than generic object detection on land, due to the overall lower image quality of underwater datasets. The degradation in image quality is caused by presence of suspended particles in the water, which introduces significant color distortion and cast. One more issue, which further complicates the task, is motion blur, caused by continuous movement of underwater vehicle.

Information Technology and Implementation (IT&I-2024), November 20-21, 2024, Kyiv, Ukraine

[*] Corresponding author.

[†] These authors contributed equally.

svm@nau.edu.ua (V. Sineglazov); mykhailo_savchenko@outlook.com (M. Savchenko); mzz@kpi.ua (M. Z. Zgurovsky).

0000-0002-3297-9060 (V. Sineglazov); 0009-0004-4072-5409 (M. Savchenko); 0000-0001-5896-7466 (M. Z. Zgurovsky).

Additionally, there are substantial differences in light emission and propagation between underwater and land scenes.

To overcome these obstacles, researchers have developed a variety of advanced techniques. Common approaches typically involve implementing the series of pre-processing steps to enhance image quality before inputting it into intellectual system, designing and deploying deeper neural networks capable of extracting meaningful features from degraded inputs, and integrating specialized network blocks that enhance feature representation abilities. Often, combinations of these methods are used, which results in complex, multi-stage processing pipelines. While these solutions have demonstrated impressive results in improving underwater object detection, they come with a significant drawback: increased overall network complexity, which translates directly into higher hardware requirements, larger model sizes, and increased power consumption.

The complexity of these advanced models makes it infeasible to deploy them on the edge devices typically used in AUVs. Thus, there is a need for research focused on size reduction strategies specifically tailored for underwater object detectors. The ultimate goal of such research should be to develop a comprehensive methodology that enables significant reductions in the size of object detection neural networks while simultaneously preserving their accuracy and operational speed.

The successful development of such methodologies could potentially lead to the design of smaller, more efficient AUVs capable of accessing environments that are currently out of reach. Considering this, this research paper aims to explore novel approaches to network size reduction for underwater object detection models, addressing this crucial challenge and contributing to the ongoing advancement of AUV technologies and their applications in marine science, environmental monitoring, and underwater operations.

2. Related work

2.1. Object detection algorithms

Generic object detection neural network topologies development started in 2014, when R-CNN model [6] was first introduced. In that timeframe, two-stage object detection methodologies were widely used, with one network responsible for region proposals, and another network handling object localization and classification. This approach is known to provide high accuracy, but the detection speed is slow due to a large number of computations caused by redundant bounding boxes. Two-stage object detectors have undergone series of improvements, with Fast R-CNN [7] and Faster R-CNN [8] being significantly more effective than the original model, but the performance was still unsuitable for real-time applications.

Later in 2015, one-stage object detection algorithms such as Single-Shot Detector (SSD) and You Only Look Once (YOLO) [9] were introduced. In these methods, a single convolutional neural network is responsible for predicting bounding boxes across all classes simultaneously by splitting the image into S x S grid, determining the tile containing the center of an object and handling confidence score calculations within it. This approach streamlines a detection process and leads to significant performance improvements by the cost of accuracy.

To improve both speed and detection accuracy, YOLO series of object detectors have undergone a lot of updates. YOLOv2 [10] introduced batch normalization and removed dropout layers. YOLOv3 [11] used residual connections in feature extraction layers and feature pyramid network (FPN) for multi-scale feature aggregation. YOLOv4 [12] introduced cross-partial connections. YOLOv5 [13] offered automated hyperparameter search. YOLOX [14] introduced anchor-less design, decoupled classification and regression head, advanced augmentations and label assignment strategy. YOLOv7 [15] offered new layer aggregation and model scaling strategies. YOLOv8 [16] implemented new augmentation strategies and offered improvements for model light-weighting. YOLOv9 [17] uses programmable gradient information to deal with information loss when data is transmitted through network layers. YOLOv10 [18] offered dual label assignment strategy to omit non-maximum suppression strategy and introduced tweaks for lower latency.

Generally, modern versions of YOLO network share the same topology, where convolutional neural network (backbone) is responsible for feature extraction at multiple scales, feature pyramid network (neck) is used to aggregate multi-scale features, mixing contextual and detailed information, and detection head is responsible for regression and classification tasks.

2.2. Underwater object detection frameworks

To overcome constraints presented by lower quality of underwater images, unique features of underwater targets such as small size and dense location, and computational constraints, specific underwater detection frameworks are developed. Typically, these frameworks can be divided into three categories by the way they achieve efficiency improvements for detecting targets in underwater environments.

The first category are object detectors with higher feature representation abilities, reached by bigger network capacity or introduction of specific blocks for better feature extraction. Attention mechanisms are often used to enhance feature extraction capabilities of the model by ensuring that backbone layers of the network focus on more relevant features [19, 20]. Liu et al. have suggested to introduce transformer blocks into object detector backbone, based on the assumption that using heterogenous architectures enhances the variability of extracted features [21]. Other array of works focuses on data augmentation and series of pre-processing steps to reach higher object detection accuracy by raising the quality and quantity levels of input data [22]. However, these methodologies share a common problem, such as overall network complexity. Although using a larger network, extra blocks and preprocessing is useful for accuracy, the number of parameters and high latency makes it infeasible to run on edge hardware, so real-world usage of this type of frameworks is restricted with using it on pre-collected data.

The second category of underwater object detection frameworks includes models, which achieve accuracy gains by focusing on mitigating a specific issue with underwater images, such as small target size, target overlap and motion blur [23, 24]. Common approaches include enhancing feature map upsampling process, using extra classification heads for smaller objects or adding extra blocks such as attention and visual transformers [25]. Main problem of the frameworks of this type are its generalization capabilities. While the accuracy is enhanced on datasets which have the problems targeted by a specific model, the same enhancements may not be applicable for other dataset, limiting its real-world usage.

The third category includes underwater object detection methodologies, which focus on decreasing the parameter count, size and latency of the model [26, 27]. Light-weighting is typically done by replacing the feature extraction portion of the model, which in case with YOLO-based detectors, is responsible for over 50% of overall computational complexity. Backbone part of object detector are typically fully or partially replaced with a mobile or light-weight convolutional neural networks, such as FasterNet [28] or GhostNet [29]. These changes drastically reduce model size and improve processing time, making the model more feasible to be used on edge devices, such as AUV integrated hardware. However, in this case, the speed and size improvements are reached by decreasing the network capacity, which leads to worse accuracy than by using generic solutions. To mitigate this effect, researchers typically introduce extra layers to enhance the representation ability of a network, which introduces extra parameters and lessens efficiency improvements, achieved by model light-weighting.

This fact has sparked our interest in comparing the performance of light-weight underwater object detectors and finding optimal way to mitigate the performance loss, which is inevitably a side-effect of model light-weighting.

2.3. Knowledge distillation

Knowledge distillation (KD) is a highly efficient technique of boosting efficiency of a light-weight student model by forcing it to mimic the outputs of a larger pre-trained teacher model. Earliest overviews on these methodologies were formulated by Bucilua [30] in 2006, and the term "knowledge distillation" and modern concept of this process was introduced in 2015 by Hinton [31]. Later, Romero et al. improved training process and student model performance by using intermediate representations as hints [32]. Zagoruyko and Komodakis suggested to use attention transfer to boost student performance [33], while Zhang et al. offered to use multiple co-learning student models [34].

While being an efficient technique, KD process ensures best results when teacher and student models share the similar architecture (e.g. CNN-to-CNN, ViT-to-ViT). As feature heterogeneity increases in later layers of neural network, it becomes harder for a student model to improve performance. Researches by Touvron et al. [35], Hao et al. [36] are offering more efficient KD algorithms for heterogenous architectures, however, the problem is still not fully resolved and KD performance is better when teacher and student models are similar.

3. Problem statement

The problem of creating an optimal size reduction methodology for YOLO-based underwater object detectors involves building two neural networks: M_{teacher} and M_{student}. Both networks accept input samples $(X_1, y_1), (X_2, y_2), \ldots, (X_i, y_i)$, where X_i denotes the i-th RGB image matrix of dimensions $n \times n \times 3$, and y_i represents the vector of ground truth bounding box coordinates, which represent the position of the object within the image, and class labels for each object within the image. The teacher network M_{teacher}, parametrized by the weights θ_{teacher}, and the student network M_{student}, parametrized by θ_{student}, are trained to handle the following transformation:

$$f_\theta : X_i \mapsto \{(\hat{b}_{ij}, \hat{c}_{ij})\}_{j=1}^{k}, \tag{1}$$

where $\hat{b}_{ij}$ is the predicted bounding box for object j in image X_i, and $\hat{c}_{ij}$ is the predicted class probability for the object. The goal is to minimize the discrepancy between the predictions $\hat{b}_{ij}, \hat{c}_{ij}$ and the ground truth b_{ij}, c_{ij} for bounding boxes and class labels, respectively.

The loss function of M_{teacher} and M_{student}, denoted as $\mathcal{L}_{\text{YOLO}}$, is composed of regression and classification losses. CIoU [37] is used as a main bounding box regression loss, defined as:

$$\mathcal{L}_{\text{CIoU}}(b, \hat{b}) = 1 - \text{IoU}(b, \hat{b}) + \frac{\rho^2(b_c, \hat{b}_c)}{c^2} + \alpha \cdot v, \tag{2}$$

where $b, \hat{b}$ are the ground truth and predicted bounding boxes, respectively, $\text{IoU}(b, \hat{b})$ is the Intersection over Union, which measures the overlap between the predicted and ground truth bounding boxes:

$$\text{IoU}(b, \hat{b}) = \frac{|b \cap \hat{b}|}{|b \cup \hat{b}|}, \tag{3}$$

where $|b \cap \hat{b}|$ is the area of overlap, $|b \cup \hat{b}|$ is the total area covered by both boxes, $\rho(b_c, \hat{b}_c)$ is the Euclidean distance between the centroids $b_c, \hat{b}_c$, α is the weight factor, balancing the importance of aspect ratio consistency:

$$\alpha = \frac{v}{(1 - \text{IoU}(b, \hat{b})) + v}, \tag{4}$$

v measures the consistency of the aspect ratio between the predicted and ground truth boxes:

$$v = \frac{4}{\pi^2}\left(\arctan\left(\frac{w}{h}\right) - \arctan\left(\frac{\widehat{w}}{\widehat{h}}\right)\right)^2, \tag{5}$$

where w and h are the width and height of the ground truth bounding box, and $\widehat{w}, \widehat{h}$ are the width and height of the predicted bounding box.

Classification is governed by varifocal loss function (VFL), defined as:

$$\mathcal{L}_{\text{VFL}}(\text{p}, \text{q}) = \begin{cases} -q(qlog(p) + (1-q)log(1-p)) & q > 0 \\ -\alpha p^{\gamma} log(1-p) & q = 0, \end{cases} \tag{6}$$

where, p is the predicted classification score, q is the target score, α is the balancing coefficient, and γ is the penalty coefficient [38].

Total loss function can be defined as the sum of these two loss functions and distribution focal loss:

$$\mathcal{L}_{\text{YOLO}} = \lambda_{\text{CIoU}}\mathcal{L}_{\text{CIoU}} + \lambda_{\text{DFL}}\mathcal{L}_{\text{DFL}} + \lambda_{\text{VFL}}\mathcal{L}_{\text{VFL}}, \tag{7}$$

with λ_{CIoU}, λ_{DFL} and λ_{VFL} being hyperparameters, balancing the importance of each component.

To reduce the size of M_{student} while maintaining performance, knowledge distillation is used for transferring knowledge from the larger teacher model M_{teacher}. The objective of knowledge distillation is to minimize a combination of the standard YOLO loss $\mathcal{L}_{\text{YOLO}}$ and distillation loss $\mathcal{L}_{\text{KD}}$, which encourages the student network to mimic the teacher network's predictions, so the total loss for the student model is defined as follows:

$$\mathcal{L}_{\text{student}} = \alpha\mathcal{L}_{\text{YOLO}} + (1-\alpha)\mathcal{L}_{\text{KD}}, \tag{8}$$

where α controls the trade-off between the YOLO loss and the distillation loss in the total loss function.

Both M_{student} and M_{teacher} are optimized using the Stochastic Gradient Descent (SGD) algorithm with the following weight update rule:

$$\theta_{t+1} = \theta_t - \eta\nabla_{\theta}\mathcal{L}, \tag{9}$$

where η is the learning rate controlling step size, and $\nabla_{\theta}\mathcal{L}$ is the gradient of the loss function with respect to the model parameters.

4. Proposed methodology

4.1. Developing knowledge distillation algorithm for YOLO

To enable knowledge transfer from larger teacher model into light-weight student model, regression and classification components of total loss function have been modified to include distillation loss with additional weighting coefficient added to avoid learning collapse due to student model fully mimicking teacher model outputs. Additionally, temperature coefficient τ with decay strategy has been used to control the softening of logits for the classification component, allowing to regulate the amount of knowledge being distilled from a larger model by a student.

Bounding box regression loss is handled by CIoU function with additional L2 loss component, improving the consistency between teacher and student model bounding box predictions:

$$\mathcal{L}_{\text{CIoU}} = (1 - \lambda_{\text{distill}}) \cdot \left(1 - \text{CIoU}(b_{ij}, \hat{b}_{ij})\right) + \lambda_{\text{distill}} \cdot \left\| \hat{b}_{ij}^{\text{teacher}} - \hat{b}_{ij}^{\text{student}} \right\|_2^2, \tag{10}$$

where λ_{distill} is the distillation coefficient balancing the standard CIoU loss and the distillation term, $\hat{b}_{ij}^{\text{teacher}}$ and $\hat{b}_{ij}^{\text{student}}$ are the bounding boxes predicted by the teacher and student networks in that order.

Classification loss is handled with varifocal loss function (VFL), with Kullback-Leibler divergence loss (KL-loss) added as a distillation component, scaled by temperature. To encourage exploration in earlier stages of student model training, we propose temperature decay strategy, starting from higher values and linearly shifting the temperature coefficient toward 1 to focus on more confident predictions in later stages of training:

$$\begin{aligned} \mathcal{L}_{\text{VFL}} = (1 - \lambda_{\text{distill}}) \cdot \text{VFL}(y_{ij}, \hat{c}_{ij}^{\text{student}}) + \lambda_{\text{distill}} \cdot \tau(t)^2 \\ \cdot \text{KL}\left(\sigma\left(\frac{\hat{c}_{ij}^{\text{teacher}}}{\tau(t)} \right) \parallel \sigma\left(\frac{\hat{c}_{ij}^{\text{student}}}{\tau(t)} \right) \right), \end{aligned} \tag{11}$$

with $\tau(t)$ representing the decaying temperature, $\hat{c}_{ij}^{\text{teacher}}$ and $\hat{c}_{ij}^{\text{student}}$ are the class probability logits for teacher and student models, respectively.

Algorithm 1
Training YOLO student and teacher models

```
**Input:** Dataset $\mathcal{D} = \{(X_i, y_i)\}_{i=1}^N$, teacher model $M_{\text{teacher}}$, student model $M_{\text{student}}$, initial temperature $\tau_0$, decay strategy $\tau(t)$, hyperparameters $\lambda_{\text{CIoU}}$, $\lambda_{\text{DFL}}$, $\lambda_{\text{VFL}}$, $\lambda_{\text{distill}}$, maximum epochs $T_{max}$, learning rate $\alpha$, batch size $B$.
**Initialize** student model parameters $\theta_{\text{student}}$, load teacher model with parameters $\theta_{\text{teacher}}$, set initial temperature $\tau = \tau_0$.
**For each epoch** $t = 1, \dots, T_{max}$:
  shuffle training set $\mathcal{D}$;
  for each mini-batch $\mathcal{B} \subset \mathcal{D}$:
    do forward path, computing $\hat{b}_{ij}^{\text{teacher}}, \hat{c}_{ij}^{\text{teacher}}$ and $\hat{b}_{ij}^{\text{student}}, \hat{c}_{ij}^{\text{student}}$;
    compute loss $\mathcal{L}_{\text{total}}$;
    do backward pass, compute gradients $\nabla_{\theta_{\text{student}}} \mathcal{L}_{\text{total}}$;
    update $\theta_{\text{student}} \leftarrow \theta_{\text{student}} - \alpha \cdot \nabla_{\theta_{\text{student}}} \mathcal{L}_{\text{total}}$;
    update temperature $\tau(t)$.
**End of training:** Return the trained student model $M_{\text{student}}$.
**Output:** Trained student model $M_{\text{student}}$.
```

4.2. Building light-weight student model topology

To ensure optimal performance of a resulting distilled model, the student model should meet the size and computational efficiency requirement. An approach used in this paper involved light-weighting the feature extraction (backbone) layers of YOLO object detector, to reduce the number of expensive convolutional operations, which contribute a lot to a total parameter count. Feature aggregation (neck) and final output layers (head) from original YOLOv8 architecture were reused, as adding additional blocks to these parts of the network would increase the parameter count, and extra light-weighting would introduce more differences between student and teacher model, which could harm the distillation performance. In this case, backbone network consists of convolutional blocks, which are composed of 2D convolutional layer and batch normalization, followed by SiLU activation function. Each convolutional block is followed by bottleneck blocks C2f or C3, which perform convolutional operation on the input, then splits the channels, processes the resulting feature map through multiple bottlenecks (number in the name of the block represents the bottleneck layer count), ending with concatenation.

Overall architecture of object detection network is shown in Figure 1.

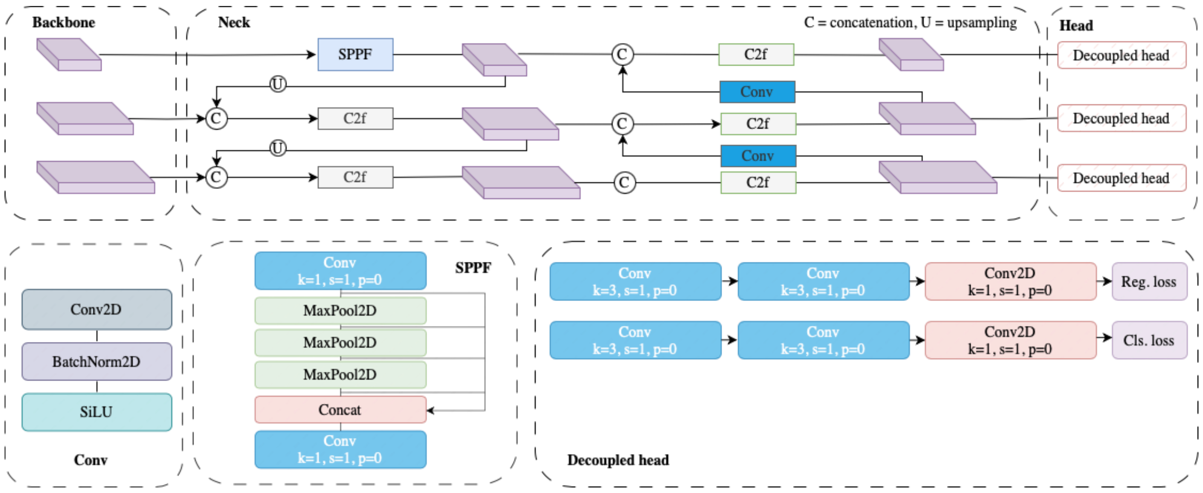

Figure 1: Overall structure of a proposed object detector.

To find the model, which provides the best balance between efficiency and performance, we have built two variants of backbone network, based on GhostConvolution layers derived from GhostNet and FasterNet blocks. The core idea behind this design is to reduce the number of computationally expensive 3 x 3 convolutions in favor of their light-weight counterparts.

Default convolution operation outputs feature map Y by processing input feature map X using a convolution kernel W as $Y = X * W$, where $*$ denotes the convolution operation. GhostConvolution is aimed at generating the similar number of feature maps by using less computations by executing convolution with fewer filters, obtaining intrinsic feature maps Y_{int} as $Y_{\mathrm{int}} = X * W_{\mathrm{int}}$ where W_{int} is a smaller convolutional filter. Then, the series of computationally inexpensive operations is used to generate the ghost feature maps Y_{ghost} as in $Y_{\mathrm{ghost}} = f(Y_{\mathrm{int}})$ where f denotes cheap operation.

FasterNet applies different approach to reduce the computational complexity and decrease latency of convolutional operations, based on PConv procedure, which applies convolutional operation only on a part of input channels for spatial feature extraction and leaves remaining channels as is. Then, PConv is followed by series of pointwise convolution to reuse the information from all channels in an efficient way.

5. Experimental evaluation

To evaluate the performance of the proposed algorithm, we used the UTDAC2020 (Underwater Target Detection Algorithm Competition 2020) dataset, a challenging underwater detection benchmark, consisting of 5168 training and 1293 validation images in various resolutions (3840 x 2160, 1920 x 1080, 720 x 405, and 586 x 480). The chosen dataset contains four classes: echinus, holothurian, scallop, and starfish. UTDAC2020 presents several challenges, including significant class imbalance, with the echinus class appearing four times more frequently than the other classes. The dataset also features targets at different scales, often densely packed, and the cases of low contrast, challenging lighting conditions and motion blur caused by camera movement.

The machine used for experiment is equipped with Intel Core i5-13600K processor, NVIDIA A4000 GPU with 16GB VRAM. Software-wise, the test setup is running Ubuntu 20.04.6 LTS with Python 3.10.13, CUDA 12.1, and PyTorch 2.2.1.

Each model has been trained during 250 epochs with a batch size of 32 and an image size of 640 x 640. Stochastic Gradient Descent (SGD) served as the optimization algorithm, with a momentum of 0.937, an initial learning rate of 0.01, and a weight decay coefficient of 0.005. For distilled models, initial temperature coefficient is set to 5, linearly decaying toward 1 until the model converges. The hyperparameters were found empirically using the Ray Tune library. Albumentations package was employed for augmenting the dataset, which includes combinations of random crop, random rotate and mosaic augmentations.

Total of five metrics have been used to test the model, with mAp and mAp_{50} representing the object detection accuracy of neural network. Size, parameter count and FLOPs are also measured as performance metrics to evaluate the computational efficiency of proposed approach. Detailed explanation of metrics is provided in Table 1.

Table 1
Metrics explanation

Metrics	Description
mAp_{50}	Mean average precision (mAp) at intersection over union (IoU) of 0.50, defined as average precision for each class over number of classes
mAp	mAp at IoU of 0.50:0.05:0.95
Params	Total number of model parameters
FLOPs	Performance metrics denoting number of floating-point operations per second
Size	Model size in megabytes

A comparison of existing frameworks and models, built using the proposed methodology, is provided in Table 2. For distilled models, YOLOv8l with DarkNet-53 backbone is used as a teacher model. Student models use YOLOv8s architecture with custom backbones, based on GhostNet and FasterNet, with both convolutional blocks and bottlenecks modified. Models using knowledge distillation are marked with '-dist' suffix.

Table 2
Experiment results on UTDAC2020 dataset

Method	Backbone	mAp	mAp_{50}	Params (M)	FLOPs (G)	Size (Mb)
Faster R-CNN	ResNet50	44.51	80.93	41.14	63.3	~
RetinaNet	ResNet50	43.93	80.42	36.17	52.6	~
FCOS	ResNet50	43.88	81.06	31.84	50.4	~
YOLOv8n	DarkNet-53	48.92	82.61	3	8.9	6
YOLOv8s	DarkNet-53	50.45	84.58	11.2	28.8	22
YOLOv8m	DarkNet-53	51.62	84.92	25.8	78.7	51
YOLOv8l	DarkNet-53	51.73	84.97	43.6	165.7	84
YOLOv8s	GhostNet	49.76	83.71	6	16.4	9
YOLOv8s	FasterNet	49.85	83.8	5.8	16	9
YOLOv8s-dist	GhostNet	50.62	84.7	6	16.4	9
YOLOv8s-dist	FasterNet	50.71	84.72	5.8	16	9

Experiment results prove superior performance of models using knowledge distillation algorithm. In comparison to YOLOv8s, the computational complexity in FLOPs is reduced by 45%, while maintaining object detection performance. The choice of light-weight network backbone didn't provide any significant difference, with FasterNet and GhostNet backbones providing similar results in terms of both size and accuracy.

Visualization of ground truth bounding boxes and class labels, compared with detections, performed with the proposed YOLOv8-dist model, is shown in Figure 2. The samples with complex background information and targets of various sizes were selected to demonstrate the performance of our model.

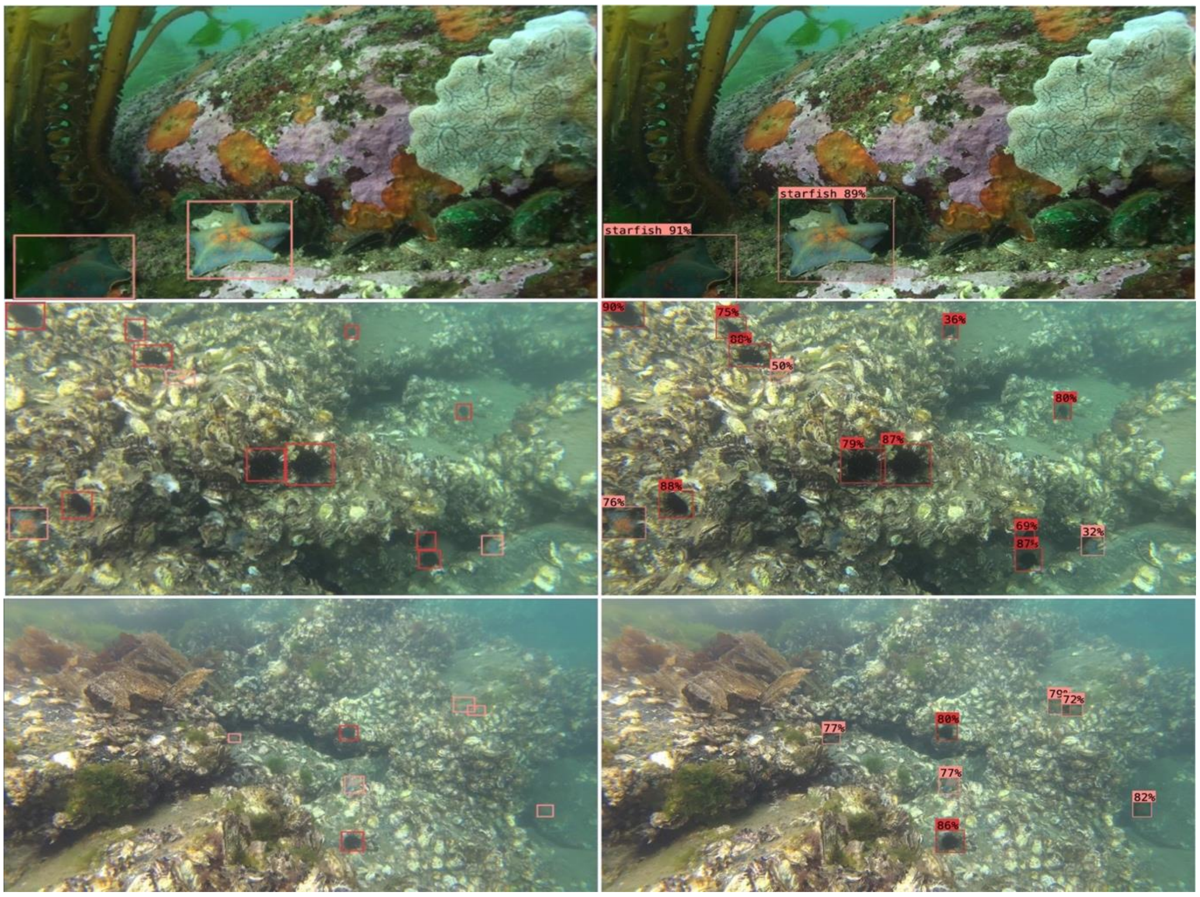

Figure 2: Examples of object detection obtained from YOLOv8s-dist model. Ground truth labels are on the left, proposed model detection results are on the right. Detection of targets at various scales and objects on complex backgrounds is handled correctly.

6. Conclusions

The paper proposed a novel methodology to reduce the size of YOLO-based underwater object detectors. Knowledge distillation algorithm with temperature decay strategy has been developed for object detection neural network, allowing to effectively train light-weight student model by transferring knowledge from teacher model of larger capacity. Additionally, we developed two light-weight YOLO architectures, derived from GhostNet and FasterNet approaches to convolution operation, which are suitable to be used as a student model in knowledge distillation tasks.

The proposed light-weight models are 45% more efficient in terms of computational complexity, compared to existing YOLOv8s model. After using our knowledge distillation algorithm, the performance of the student model is superior to original YOLOv8s in terms of accuracy (84.72% and 84.58%, respectively), while the model size is comparable to YOLOv8n, the smallest model among YOLO-based detectors.

The applications of our methodology include training efficient object detection neural network for integrated autonomous underwater vehicle hardware. To achieve even better performance of knowledge distillation with YOLO-based detectors, we suggest that further research could be conducted on more sophisticated knowledge distillation techniques during training, and using knowledge distillation algorithms with heterogenous backbones, such as visual transformers, which could potentially enrich intermediate feature maps with more context and semantic information, leading to a higher accuracy in applied tasks.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] M. Zgurovsky, V. Sineglazov, E. Chumachenko, Classification and Analysis Topologies Known Artificial Neurons and Neural Networks, in: Studies in Computational Intelligence, Springer International Publishing, Cham, 2020, pp. 1–58. doi:10.1007/978-3-030-48453-8_1.

[2] V. M. Sineglazov, K. D. Riazanovskiy, O. I. Chumachenko, Multicriteria conditional optimization based on genetic algorithms, Syst. Res. Inf. Technol. No. 3 (2020) 89–104. doi:10.20535/srit.2308-8893.2020.3.07.

[3] V. Sineglazov, A. Kot, Design of Hybrid Neural Networks of the Ensemble Structure, SSRN Electron. J. (2021). doi:10.2139/ssrn.3807474.

[4] M. Zgurovsky, V. Sineglazov, E. Chumachenko, Formation of Hybrid Artificial Neural Networks Topologies, in: Studies in Computational Intelligence, Springer International Publishing, Cham, 2020, pp. 175–232. doi:10.1007/978-3-030-48453-8_3.

[5] M. Zgurovsky, V. Sineglazov, E. Chumachenko, Development of Hybrid Neural Networks, in: Studies in Computational Intelligence, Springer International Publishing, Cham, 2020, pp. 233–312. doi:10.1007/978-3-030-48453-8_4.

[6] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2014. doi:10.1109/cvpr.2014.81.

[7] R. Girshik, Fast R-CNN, 2015. URL: https://arxiv.org/abs/1504.08083.

[8] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, IEEE Trans. Pattern Anal. Mach. Intell. 39.6 (2017) 1137–1149. doi:10.1109/tpami.2016.2577031.

[9] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016. doi:10.1109/cvpr.2016.91.

[10] J. Redmon, A. Farhadi, YOLO9000: Better, Faster, Stronger, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017. doi:10.1109/cvpr.2017.690.

[11] J. Redmon, A. Farhadi, YOLOv3: An Incremental Improvement, 2018. URL: http://arxiv.org/abs/1804.02767v1.

[12] A. Bochkovskiy, C.-Y. Wang, H.-Y. M. Liao, YOLOv4: Optimal Speed and Accuracy of Object Detection, 2020. URL: https://arxiv.org/abs/2004.10934.

[13] Ultralytics, GitHub - ultralytics/yolov5: YOLOv5 in PyTorch > ONNX > CoreML > TFLite, 2020. URL: https://github.com/ultralytics/yolov5.

[14] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, et al., YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications, 2022. URL: https://arxiv.org/abs/2209.02976.

[15] C.-Y. Wang, A. Bochkovskiy, H.-Y. M. Liao, YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors, in: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2023. doi:10.1109/cvpr52729.2023.00721.
[16] Ultralytics, Ultralytics YOLO Docs, 2023. URL: https://docs.ultralytics.com.
[17] C.-Y. Wang, I.-H. Yeh, H.-Y. M. Liao, YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information, 2024. URL: https://arxiv.org/abs/2402.13616.
[18] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, G. Ding, YOLOv10: Real-Time End-to-End Object Detection, 2024. URL: https://arxiv.org/abs/2405.14458.
[19] W. Zhou, F. Zheng, G. Yin, Y. Pang, J. Yi, YOLOTrashCan: A Deep Learning Marine Debris Detection Network, IEEE Trans. Instrum. Meas. (2022) 1. doi:10.1109/tim.2022.3225044.
[20] L. Ren, Z. Li, X. He, L. Kong, Y. Zhang, An Underwater Target Detection Algorithm Based on Attention Mechanism and Improved YOLOv7, Comput., Mater. & Contin. 78.2 (2024) 2829–2845. doi:10.32604/cmc.2024.047028.
[21] K. Liu, L. Peng, S. Tang, Underwater Object Detection Using TC-YOLO with Attention Mechanisms, Sensors 23.5 (2023) 2567. doi:10.3390/s23052567.
[22] J.-M. Noh, G.-R. Jang, K.-N. Ha, J.-H. Park, Data Augmentation Method for Object Detection in Underwater Environments, in: 2019 19th International Conference on Control, Automation and Systems (ICCAS), IEEE, 2019. doi:10.23919/iccas47443.2019.8971728.
[23] S. Qu, C. Cui, J. Duan, Y. Lu, Z. Pang, Underwater small target detection under YOLOv8-LA model, Sci. Rep. 14.1 (2024). doi:10.1038/s41598-024-66950-w.
[24] Y. Sun, W. Zheng, X. Du, Z. Yan, Underwater Small Target Detection Based on YOLOX Combined with MobileViT and Double Coordinate Attention, J. Mar. Sci. Eng. 11.6 (2023) 1178. doi:10.3390/jmse11061178.
[25] M. Zhang, Z. Wang, W. Song, D. Zhao, H. Zhao, Efficient Small-Object Detection in Underwater Images Using the Enhanced YOLOv8 Network, Appl. Sci. 14.3 (2024) 1095. doi:10.3390/app14031095.
[26] A. F. Ayob, K. Khairuddin, Y. M. Mustafah, A. R. Salisa, K. Kadir, Analysis of Pruned Neural Networks (MobileNetV2-YOLO v2) for Underwater Object Detection, in: Lecture Notes in Electrical Engineering, Springer Singapore, Singapore, 2020, pp. 87–98. doi:10.1007/978-981-15-5281-6_7.
[27] M. Zhang, S. Xu, W. Song, Q. He, Q. Wei, Lightweight Underwater Object Detection Based on YOLO v4 and Multi-Scale Attentional Feature Fusion, Remote Sens. 13.22 (2021) 4706. doi:10.3390/rs13224706.
[28] J. Chen, S.-h. Kao, H. He, W. Zhuo, S. Wen, C.-H. Lee, S. H. G. Chan, Run, Don't Walk: Chasing Higher FLOPS for Faster Neural Networks, in: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2023. doi:10.1109/cvpr52729.2023.01157.
[29] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, C. Xu, GhostNet: More Features From Cheap Operations, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2020. doi:10.1109/cvpr42600.2020.00165.
[30] C. Buciluǎ, R. Caruana, A. Niculescu-Mizil, Model compression, in: the 12th ACM SIGKDD international conference, ACM Press, New York, New York, USA, 2006. doi:10.1145/1150402.1150464.
[31] G. Hinton, O. Vinyals, J. Dean, Distilling the Knowledge in a Neural Network, 2015. URL: https://arxiv.org/abs/1503.02531.
[32] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, Y. Bengio, FitNets: Hints for Thin Deep Nets, 2014. URL: https://arxiv.org/abs/1412.6550.
[33] S. Zagoruyko, N. Komodakis, Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer, 2016. URL: https://arxiv.org/abs/1612.03928.

[34] Y. Zhang, T. Xiang, T. M. Hospedales, H. Lu, Deep Mutual Learning, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2018. doi:10.1109/cvpr.2018.00454.

[35] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jegou, Training data-efficient image transformers & distillation through attention, in: Proceedings of the 38th International Conference on Machine Learning, Journal of Machine Learning Research, 2021, pp. 10347–10357. URL: https://proceedings.mlr.press/v139/touvron21a.html.

[36] Z. Hao, J. Guo, K. Han, Y. Tang, H. Hu, Y. Wang, C. Xu, One-for-All: Bridge the Gap Between Heterogeneous Architectures in Knowledge Distillation, 2023. URL: https://arxiv.org/abs/2310.19444.

[37] Z. Zheng, P. Wang, D. Ren, W. Liu, R. Ye, Q. Hu, W. Zuo, Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation, 2020. URL: https://arxiv.org/abs/2005.03572.

[38] H. Zhang, Y. Wang, F. Dayoub, N. Sunderhauf, VarifocalNet: An IoU-aware Dense Object Detector, in: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2021. doi:10.1109/cvpr46437.2021.00841.