


Article

A Pruning and Distillation Based Compression Method for Sonar Image Detection Models

Chensheng Cheng, Xujia Hou, Can Wang, Xin Wen, Weidong Liu and Feihu Zhang * 

School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an 710072, China; chensheng.cheng@mail.nwpu.edu.cn (C.C.); hxj1363947894@mail.nwpu.edu.cn (X.H.); can.wang@mail.nwpu.edu.cn (C.W.); wenxin666@mail.nwpu.edu.cn (X.W.); liuwd@nwpu.edu.cn (W.L.)

* Correspondence: feihu.zhang@nwpu.edu.cn

Abstract: Accurate underwater target detection is crucial for the operation of autonomous underwater vehicles (AUVs), enhancing their environmental awareness and target search and rescue capabilities. Current deep learning-based detection models are typically large, requiring substantial storage and computational resources. However, the limited space on AUVs poses significant challenges for deploying these models on the embedded processors. Therefore, research on model compression is of great practical importance, aiming to reduce model parameters and computational load without significantly sacrificing accuracy. To address the challenge of deploying large detection models, this paper introduces an automated pruning method based on dependency graphs and successfully implements efficient pruning on the YOLOv7 model. To mitigate the accuracy degradation caused by extensive pruning, we design a hybrid distillation method that combines output-based and feature-based distillation techniques, thereby improving the detection accuracy of the pruned model. Finally, we deploy the compressed model on an embedded processor within an AUV to evaluate its performance. Multiple experiments confirm the effectiveness of our proposed method in practical applications.

Keywords: pruning; knowledge distillation; sidescan sonar; object detection; YOLOv7; embedded device



Citation: Cheng, C.; Hou, X.; Wang, C.; Wen, X.; Liu, W.; Zhang, F. A Pruning and Distillation Based Compression Method for Sonar Image Detection Models. *J. Mar. Sci. Eng.* **2024**, *12*, 1033. <https://doi.org/10.3390/jmse12061033>

Academic Editor: Sergei Chernyi

Received: 21 May 2024

Revised: 10 June 2024

Accepted: 16 June 2024

Published: 20 June 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In the field of object detection, deep learning has achieved tremendous success due to its powerful feature extraction capabilities, effectively driving the development of this domain. Currently, convolutional neural network based object detection models have been improving in performance as their network depth increases, as seen in RCNN [1,2], SSD [3,4], and YOLO [5,6]. However, as the network depth of these models continues to grow, the storage space required for the models and the computational resources needed for inference also increase, posing challenges for specific tasks that cannot utilize high-performance processors. Consequently, many researchers have begun to investigate how to ensure that models perform well on edge devices with limited computational resources [7,8].

Accurate target detection capabilities are very important for autonomous underwater vehicles (AUVs) performing underwater operations, but model deployment is constrained by computing resources [9,10]. The core devices responsible for data storage, data processing, system decision making, and power management are housed within a watertight electronics compartment. Due to the carrying capacity and overall space constraints of the AUV, the volume and weight of the watertight electronics compartment are limited, which prevents the use of large, high-performance processors. Therefore, researching network compression methods to strike a balance between detection accuracy and model size is crucial. This research aims to enable the successful deployment of powerful object detection networks on underwater embedded devices with limited computing and memory capacities, representing a highly significant practical problem worth investigating.

In the realm of model compression for deep neural networks, six primary research directions can be delineated: network pruning [11], lightweight network design [12], neural architecture search (NAS) [13], network quantization [14], knowledge distillation (KD) [15], and low-rank decomposition [16]. Network pruning involves deleting redundant parameters deemed unimportant through various evaluation criteria, leading to sparsity and consequent model compression. This method significantly reduces model size, lowers computational complexity, and enhances operational efficiency. In recent years, network pruning has gained popularity. Lightweight network design focuses on directly designing more compact convolutional kernels or methods to reduce the number of network structure parameters, thereby reducing model size and computational complexity. However, this may necessitate expertise to design effective lightweight network structures. NAS automates the search for optimal network architectures, improving model performance and efficiency. However, this method consumes significant computational resources, has a vast search space, and requires extensive training time and data, making it less accessible to general researchers. Network quantization converts high-precision operations into low-precision operations (floating-point to integer operations), reducing model storage space and computational load, and accelerating inference speed. However, quantization may sacrifice some model accuracy, requiring careful selection of quantization methods and parameters. KD transfers knowledge learned by a complex, parameter-heavy teacher network to a relatively simpler student network, achieving similar expressive capabilities while saving time and space costs. Due to its ability to significantly enhance small model performance, KD has become one of the mainstream methods in model compression. Low-rank decomposition aims to decompose convolutional kernels using matrix analysis methods like low-rank decomposition, approximating the original weight matrix by decomposing large convolutional kernel tensors into smaller ones, thus reducing the computational and parameter requirements of the model to some extent. However, implementing low-rank decomposition methods is not straightforward and involves higher computational costs.

In the field of underwater acoustic target detection, researchers have faced challenges due to scarce datasets and experimental difficulties, resulting in relatively limited studies on network compression. Tian et al. [17] optimized the YOLOv4 model using channel and layer pruning methods and successfully deployed it on underwater vehicles for target detection. It is worth noting that the pruned model retained only 7 % (from 245 M to 17 M) of the original parameters, ensuring real-time requirements for edge devices. Szymak et al. [18] adopted a more effective pre-training method to deploy the deep learning model on the Nvidia Jetson TX2 and realized target classification based on underwater videos. Ye et al. [19] introduced the Mobilenetv3 architecture into the backbone network of YOLOv4 to lightweight the network, reducing the model's parameter size to a quarter of the original, significantly alleviating storage pressure on edge devices. Yan et al. [20] improved the MobileNet-SSD benchmark network for underwater sonar target detection, greatly enhancing the network's inference speed. Hu et al. [21] integrated ShuffleNetv2 into YOLOv5s to make the model more compact, improving computational speed on embedded devices. Qin et al. [22] pruned the improved YOLOv7 using channel pruning methods, reducing model memory by 47.50% and increasing detection speed by nearly 2.5 times for forward-looking sonar target detection.

After considering usability, cost, and practical engineering demands, this study ultimately employed a combined approach of pruning and distillation for model compression. We focused on the YOLOv7 network model and successfully achieved substantial compression of the object detection network while maintaining detection accuracy. Furthermore, we validated the real-time performance and robustness of this lightweight network using both publicly available datasets and data we collected ourselves.

The main contributions of this paper are as follows:

- (1) Addressing the challenge of deploying overly bulky detection models on edge devices, we introduce a dependency graph-based pruning method and efficiently implement pruning on the YOLOv7 model.

- (2) To counter the significant drop in model accuracy after extensive pruning, we devise a fusion distillation method that combines output-based distillation with feature-based distillation, thereby enhancing the detection accuracy of pruned models.
- (3) Deploying the compressed models on edge devices for testing, we validate the real-time detection performance of the methods utilized in this paper on embedded devices.

2. Related Work

2.1. Network Pruning

Currently, mainstream pruning methods can be divided into two categories: non-structural pruning and structural pruning. Non-structural pruning does not alter the network's structure but rather zeros out some of the weights. However, the distribution of these zeros is generally irregular, requiring specialized hardware and software acceleration for proper functioning. In contrast, structural pruning modifies the neural network's structure by physically removing grouped parameters based on predefined rules. This approach does not rely on specific AI accelerators or software, making it more widely applicable in engineering projects than non-structural pruning.

In the realm of non-structural pruning, the origins can be traced back to 1989 when LeCun et al. [23] balanced network complexity and training set errors using second-order derivative information to remove unimportant weights from the network. Their aim was to achieve a network with better generalization and improved inference speed. Subsequently, Han et al. [24] proposed a non-structured pruning method based on threshold weight values, analyzing weight parameters to prune the network structure. However, setting pruning thresholds requires a strong mathematical background and manual intervention, severely limiting its engineering application. In response to this challenge, Li et al. [25] introduced an optimization-based method for automatically adjusting thresholds, transforming the threshold adjustment problem into a constrained optimization problem and solving it using derivative-free optimization algorithms. Lee et al. [26] highlighted the importance of global pruning by studying hierarchical sparsity, introducing an adaptive magnitude pruning score that does not rely on manual hyperparameter tuning. Moreover, determining the importance of weights scientifically is a major research focus. Carreira et al. [27], for instance, framed pruning as an optimization problem, minimizing the loss of weights while satisfying basic pruning conditions and automatically learning the optimal number of weights to prune in each layer. Kwon et al. [28] proposed a novel representation scheme for sparsely quantized neural network weights, achieving high compression ratios through fine-grained and non-structural pruning while maintaining model accuracy.

In the realm of structural pruning, methods can be broadly categorized into neuron pruning, filter pruning, channel pruning, and layer pruning based on the granularity of pruning. Molchanov et al. [29] approximated neurons' contribution to the loss function using first- and second-order Taylor expansions and iteratively removed neurons with low scores. Wang et al. [30] proposed a layer-adaptive filter pruning method based on reducing structural redundancy, constructing a graph for each convolutional layer in CNNs to measure their redundancy. This method prunes unimportant filters in the most redundant layers rather than in all layers. To address the additional hyperparameters and long training cycles associated with filter pruning methods, Ruan et al. [31] introduced a novel dynamic and progressive filter pruning scheme (DPFPS), solving the optimization problem based on pruning ratios with an iterative soft-thresholding algorithm that exhibits dynamic sparsity. At the end of training, only redundant parameters need to be removed without additional stages. Ding et al. [32] proposed a lossless channel pruning method called ResRep, reducing CNN size by narrowing convolutional layer widths. However, premature pruning of some channels can severely impact model accuracy. To mitigate this issue, Hou et al. [33] suggested iteratively pruning and regrowing channels throughout the entire training process, reducing the risk of prematurely pruning important channels. Yang et al. [34] discovered that merely reducing CNN model size or computational complexity does not necessarily reduce energy consumption. To bridge the gap between CNN design and

energy optimization, they proposed a CNN energy-aware pruning algorithm that uses CNN energy consumption directly to guide layer pruning processes.

2.2. Knowledge Distillation

Currently, knowledge distillation can be classified into three types based on the type of knowledge: output-based distillation, feature-based distillation, and relation-based distillation. Researchers need to choose the distillation method based on the specific task and may also consider combining different distillation methods for optimal results.

Buciluă et al. [35] were the first to propose a method for training fast and compact small models to mimic the performance of larger, slower, and more complex models. They achieved this by training the small model using data obtained from inference of the large model. Their method resulted in minimal performance loss for the small model, demonstrating that knowledge acquired from a large ensemble of models can be transferred to simpler, smaller models. Building upon this, Hinton et al. [36] introduced a more general solution termed “knowledge distillation”. They defined soft targets (class probability distribution generated by the teacher network’s Softmax layer) and hard targets (one-hot encoding of true labels) and introduced a “temperature” variable to control the smoothness of the class probability distribution. The student model was then trained to learn both soft and hard targets simultaneously, with the weighted average of the two target loss functions serving as the overall loss function, thereby improving the performance of the student model. Subsequently, Zhang et al. [37] proposed a deep mutual learning method that allows the student model to learn knowledge from multiple teacher models. To enhance the student model’s understanding of the teacher models, Kim et al. [38] introduced a method of encoding and decoding the teacher network’s outputs, leading to improved results for the student learning the decoded knowledge. In order to fully utilize the outputs of the teacher network, Zhao et al. [39] proposed a decoupled knowledge distillation (DKD) method, dividing the distillation loss into target class loss and non-target class loss, significantly enhancing the distillation performance.

The mentioned methods are all based on output distillation, yet some researchers argue that response-based distillation has certain limitations and propose feature-based distillation methods. Adriana et al. [40] argued that for deep neural networks, it is challenging to propagate supervisory signals all the way back to the front end, and constraining only at the label level is insufficient. Therefore, they propose the Fitnet method, introducing supervisory signals in the middle hidden layers of the network, using hints training to constrain the outputs of the intermediate layers of the two models to be as close as possible, allowing the student model to learn the intermediate layer features of the teacher model. Zagoruyko et al. [41] proposed attention transfer methods, enabling the student to learn the attention distribution of the teacher to enhance model performance. Huang et al. [42] introduced the neuron selectivity transfer (NST) method, enabling the student model to better mimic the neuron activation patterns of the teacher model, thereby improving knowledge distillation effectiveness. Ahn et al. [43] presented an information-theoretic framework for knowledge transfer, formulating knowledge transfer as maximizing mutual information between teacher and student networks, which has significant advantages in cross-heterogeneous network architecture knowledge transfer. To simplify the student model’s learning process, Yang et al. [44] used an adaptive instance normalization method to perform feature statistics on each feature channel’s data, making it easier for the student to capture the teacher model’s higher-level feature representations.

In addition to output-based and feature-based distillation methods, some researchers believe that it is crucial for the student model to learn the relationships between network feature layers, known as relation-based distillation. Yim et al. [45] considered that a true teacher teaches the process of problem-solving, so they redefine knowledge distillation as learning the process of solving problems, specifically focusing on learning the relationships between network feature layers. They distill knowledge by fitting the relationships between teacher and student layers, allowing the student to learn the process of feature extraction

from the teacher model. Subsequently, Peng et al. [46] proposed a distillation method based on correlation congruence, transferring knowledge by comparing the correlation of feature representations between the student and teacher models, aiming to align the correlation of the student model with that of the teacher model. Park et al. [47] introduced a relationship knowledge distillation (RKD) method, proposing distillation losses based on distance and angle constraints to transfer the mutual relationships between data instances. To further refine the knowledge of the teacher model, Liu et al. [48] introduced an instance relationship graph (IRG)-based method, modeling instance features, instance relationships, and feature space transformations, allowing the student's IRG to mimic the structure of the teacher's IRG. Zhou et al. [49] designed a student model based on graph neural networks (GNNs), leveraging GNNs to capture and transfer the global knowledge of the teacher model, effectively improving the performance of the student model.

In summary, unstructured pruning methods can achieve large-scale model pruning but require specialized hardware support for deployment. Current structured pruning methods are mostly designed for specific network architectures and necessitate manually crafted pruning rules, which are labor-intensive and less generalizable. In the realm of knowledge distillation, methods based on outputs and features are easier to implement, whereas those based on relationships are more complex and challenging to integrate with other distillation techniques. Additionally, due to the high cost and difficulty of underwater acoustic experiments, there remains a significant research gap in model compression for underwater acoustic target detection. Therefore, this paper employs an automated pruning scheme based on dependency graphs, combined with output- and feature-based distillation methods. This approach achieves substantial model compression while maintaining detection accuracy and has been validated for performance on embedded devices in underwater environments.

3. Methods

This study takes into account the characteristics of dense object detectors. Firstly, an automatic pruning method based on dependency graphs is employed to prune the YOLOv7 model. Subsequently, a combination of output-based binary cross-entropy classification distillation and feature-based mask generation distillation methods is applied to distill the pruned model, aiming to enhance its detection performance.

3.1. Automated Pruning Method Based on Dependency Graphs

Pruning is a highly challenging task for modern deep neural networks, given the complex interconnections between various modules in neural networks, as illustrated in Figure 1. When removing a channel, one must simultaneously consider its interdependencies with other layers; otherwise, we will end up with a damaged network. Additionally, manually analyzing dependency patterns for each module and designing pruning schemes individually is cumbersome and not conducive to widespread adoption. Therefore, this paper introduces a pruning method based on dependency graphs [50], which automatically constructs dependency relationships, groups all parameters according to these dependencies, uniformly evaluates the importance of grouped parameters, and ultimately achieves efficient pruning.

In addressing the complex dependencies among the modules of the detection network, we decompose the entire network model $\mathcal{M}(x; w)$ into more detailed components by layer. The network model can be represented as $\mathcal{M} = \{f_1, f_2, \dots, f_L\}$, where f can be convolution layers, residual operations, activation functions, and batch normalization (BN) layers, among others. The inputs and outputs of the f_i layer are denoted as f_i^- and f_i^+ , respectively, allowing for the use of different pruning strategies for the input and output parts of the same layer. Ultimately, the entire neural network can be expressed using Equation (1):

$$\mathcal{M} = \{f_1^-, f_1^+, f_2^-, f_2^+, \dots, f_L^-, f_L^+\} \quad (1)$$

At this point, the dependencies within the entire neural network can be categorized into intra-layer dependencies and inter-layer dependencies, which can be represented using Equation (2):

$$(f_1^-, f_1^+) \leftrightarrow \underbrace{(f_2^-, f_2^+)}_{\text{Inter-layer Dep}} \cdots \leftrightarrow \underbrace{(f_L^-, f_L^+)}_{\text{Intra-layer Dep}} \quad (2)$$

Inter-layer dependencies always exist, while intra-layer dependencies exist only when the input and output share the same pruning scheme. For instance, the BN layer exhibits intra-layer dependencies, whereas the convolutional layer does not. According to these rules, we can build a dependency model, denoted as Equation (3):

$$D(f_i^-, f_j^+) = \underbrace{L[i - j = 1 \wedge f_i^- \leftrightarrow f_j^+]}_{\text{Inter-layerDep}} \vee \underbrace{L[i = j \wedge sch(f_i^-) = sch(f_j^+)]}_{\text{Intra-layerDep}} \quad (3)$$

In Equation (3), the symbols \vee and \wedge represent logical “or” and “and” operations, respectively. L is a decision function. When the inputs are adjacent layers, a dependency always exists, and L outputs “True”. When the inputs are within the same layer and share the same pruning strategy, a dependency also exists, and L outputs “True”. However, when the inputs are within the same layer but use different pruning strategies, no dependency exists, and L outputs “False”. $sch(f_i^-)$ and $sch(f_j^+)$ respectively represent pruning schemes used for layers f_i^- and f_j^+ . The first term checks for inter-layer dependencies caused by network connections, while the second term examines intra-layer dependencies introduced by shared pruning schemes between layer inputs and outputs. It is worth noting that $D(f_i^-, f_j^+)$ is a symmetric matrix, where $D(f_i^-, f_j^+) = D(f_j^+, f_i^-)$. Therefore, we can examine all input–output pairs to obtain the network’s dependency graph.

After constructing the complete dependency graph, another challenge that arises is the group-level pruning problem. A common method for assessing the importance of group-level parameters is to compute an aggregate score. However, due to differences in data distribution and magnitude between different layers, the importance score of group parameters cannot be simply obtained by summing up the numerical values of each layer. Therefore, in the process of sparsification training, we introduce a simple regularization term for each parameter $w[k]$ with k prunable dimension, as shown in Equation (4):

$$\mathcal{R}(g, k) = \sum_{k=1}^K \gamma_k \cdot I_{g,k} = \sum_{k=1}^K \sum_{w \in g} \gamma_k \|w[k]\|_2^2 \quad (4)$$

Here, g represents a group of parameters with dependencies and can be expressed as $g = \{w_1, w_2, \dots, w_g\}$, $I_{g,k} = \sum_{w \in g} \|w[k]\|_2^2$ represents the importance of the k prunable dimension, while γ_k represents the shrinkage strength applied to these parameters, which can be represented by Equation (5):

$$\gamma_k = 2^{\alpha(I_g^{\max} - I_{g,k}) / (I_g^{\max} - I_g^{\min})} \quad (5)$$

The value of γ_k ranges between $[2^0, 2^\alpha]$. In the models and datasets used in this study, we found that the pruning performance was optimized when $\alpha = 2$. Therefore, in our experiments, the value of this parameter was set to 2. Based on the model construction and regularization methods mentioned above, we conducted sparsification training on the YOLOv7 model, removing unimportant parameters, and ultimately obtained a pruned lightweight network.

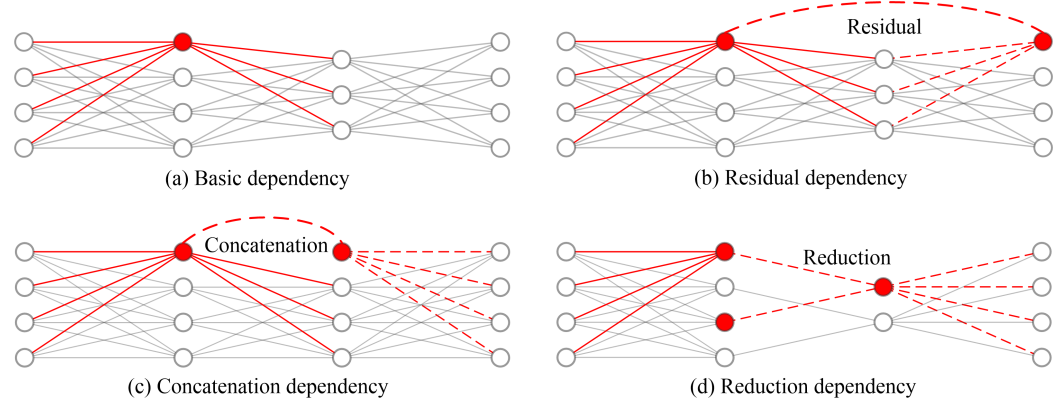


Figure 1. Complex dependencies in network models. In the figure, the solid red lines represent parameters directly connected to pruned neurons, while the dashed red lines indicate a special dependency on the pruned neurons, leading to the pruning of some indirect parameters.

3.2. Output Distillation Method Based on Binary Cross-Entropy and IoU

After pruning, the detection performance of the network tends to decrease, often requiring fine-tuning to restore its detection accuracy. However, the recoverable accuracy is usually limited. Therefore, this study employs an output distillation method [51] tailored for dense detectors like YOLOv7 to enhance the detection accuracy of the pruned model. The specific distillation method is illustrated in Figure 2.

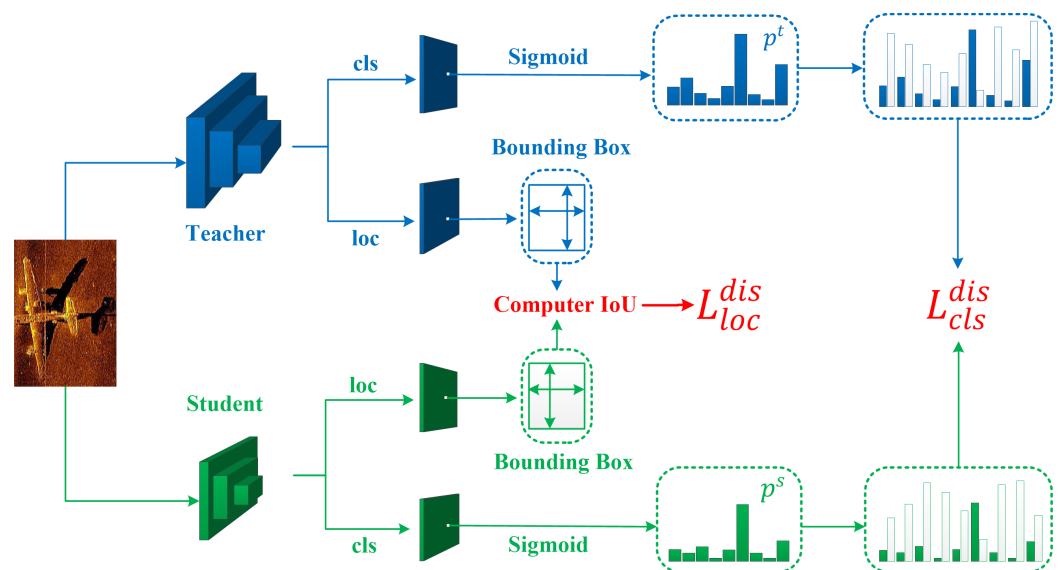


Figure 2. Schematic diagram of the method based on output distillation. For the task of object detection, separate loss functions are designed for classification and localization. In terms of classification distillation, the Sigmoid function is used instead of the Softmax function to process the classification logit map, and then the distillation loss is calculated based on binary cross-entropy. For localization distillation, the distillation loss function is constructed by calculating the IoU between the two models.

Dense detectors like YOLOv7 utilize the Sigmoid function for classification tasks in object detection, whereas the Softmax function is used in classification distillation tasks. This leads to a problem of inconsistent cross-task protocols in dense object detection. Specifically, when the classification distillation loss equals 0, there is an inconsistency between the scores of the student model and the teacher model in dense object detection, as illustrated in Figure 3.

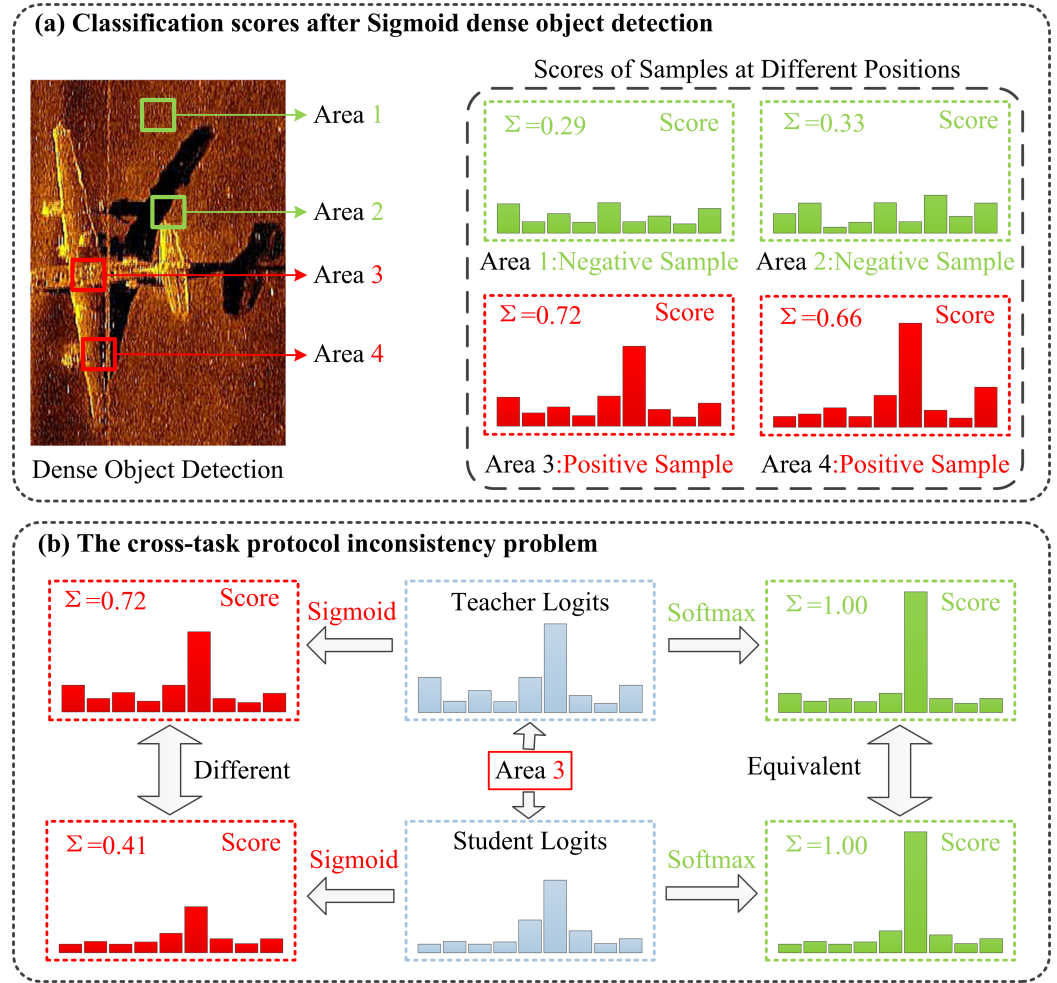


Figure 3. (a) In dense detectors like YOLOv7, the classification scores of different samples vary, which differs from image classification based on the Softmax function. (b) There is a cross-task protocol inconsistency issue. When the classification logit maps of a disparate teacher model and student model are processed using Softmax, they might produce identical scores, causing the student model to stop learning. However, using the Sigmoid function preserves the differences between them, allowing the student model to continue learning from the teacher model.

To address this issue, this study employs a binary cross-entropy distillation loss constructed from the classification scores of the teacher model and the student model. For dense detectors, an image is segmented into n regions. Assuming there are K classes, after passing through the Sigmoid function, we obtain $n * K$ classification scores. Let x be a sample, and p be the classification scores after Sigmoid activation. We can calculate the binary cross-entropy loss for the i -th feature map in the teacher and student networks for the j -th class as Equation (6):

$$\mathcal{L}_{BCE}(p_{i,j}^s, p_{i,j}^t) = -\left((1 - p_{i,j}^t) \cdot \log(1 - p_{i,j}^s) + p_{i,j}^t \cdot \log(p_{i,j}^s)\right) \quad (6)$$

The $p_{i,j}^t$ represents the classification scores of the teacher network, while $p_{i,j}^s$ represents the classification scores of the student network. Based on the binary cross-entropy loss function mentioned above, the distillation loss for the class can be calculated using Equation (7):

$$\mathcal{L}_{cls}^{dis}(x) = \sum_{i=1}^n \sum_{j=1}^K \mathcal{L}_{BCE}(p_{i,j}^s, p_{i,j}^t) \quad (7)$$

Additionally, to enhance attention on important samples, this study introduces a weighted loss strategy for computing importance weight w , which can be represented as Equation (8):

$$w = |p^t - p^s| \tag{8}$$

The final class distillation loss can be represented as Equation (9):

$$\mathcal{L}_{cls}^{dis}(x) = \sum_{i=1}^n \sum_{j=1}^K w_{i,j} \cdot \mathcal{L}_{BCE}(p_{i,j}^s, p_{i,j}^t) \tag{9}$$

In terms of localization distillation, it is generally done by converting predicted boxes into probability distributions using the Softmax function, and then constructing a loss function between the teacher network and student network for distillation. This method requires the use of discrete position probability prediction heads, which are not commonly used in current dense detection models (such as the YOLO series of detectors). Therefore, in terms of localization, this paper introduces an unstructured localization distillation loss function based on intersection over union (IoU), which is driven by the IoU loss widely used in dense object detectors, to replace existing localization distillation losses.

Specifically, we obtain the information of detection boxes from the teacher model and student model. For a given input sample x , the predicted boxes on the i -th feature map of the teacher network and student network are denoted as b_i^t and b_i^s , respectively. Then, the IoU loss between them, denoted as u_i , can be calculated, and the localization loss can be expressed as Equation (10):

$$\mathcal{L}_{loc}^{dis}(x) = \sum_{i=1}^n (1 - u_i) \tag{10}$$

The final loss function based on the logic distillation part can be expressed as Equation (11):

$$\mathcal{L}_{logits}^{dis}(x) = \beta_1 \cdot \mathcal{L}_{cls}^{dis}(x) + \beta_2 \cdot \mathcal{L}_{loc}^{dis}(x) \tag{11}$$

where β_1 and β_2 represent the weights of classification distillation loss and localization distillation loss, respectively.

3.3. Feature Distillation Method Based on Mask Generation

The feature-based distillation method typically involves encouraging the student model to mimic the outputs of the teacher model as closely as possible, as the teacher model often has stronger feature representation capabilities. However, recent studies have shown that a distillation method forcing the student model to reconstruct the output features of the teacher model using partial pixels can enhance the feature representation capability of the student model more effectively than fully mimicking the outputs of the teacher model. Additionally, this approach brings the detection accuracy of the student model closer to that of the teacher model. Therefore, in this study, we employ a feature distillation method based on mask generation to distill the intermediate feature maps of the network model [52], as illustrated in Figure 4.

Firstly, we denote the feature maps of the l -th layer of the teacher model and the student model as $T^l \in R^{C \times H \times W}$ and $S^l \in R^{C \times H \times W}$ ($l = 1, \dots, L$), respectively. Then, we set a random mask for the l -th layer feature map $M_{i,j}^l$:

$$M_{i,j}^l = \begin{cases} 0, & \text{if } R_{i,j}^l < \lambda \\ 1, & \text{Otherwise} \end{cases} \tag{12}$$

Here, $R_{i,j}^l$ represents the feature value at the i -th row and j -th column of the l -th layer feature map. Due to normalization, the values of $R_{i,j}^l$ range from 0 to 1. λ is a tunable hyperparameter that can represent the mask ratio.

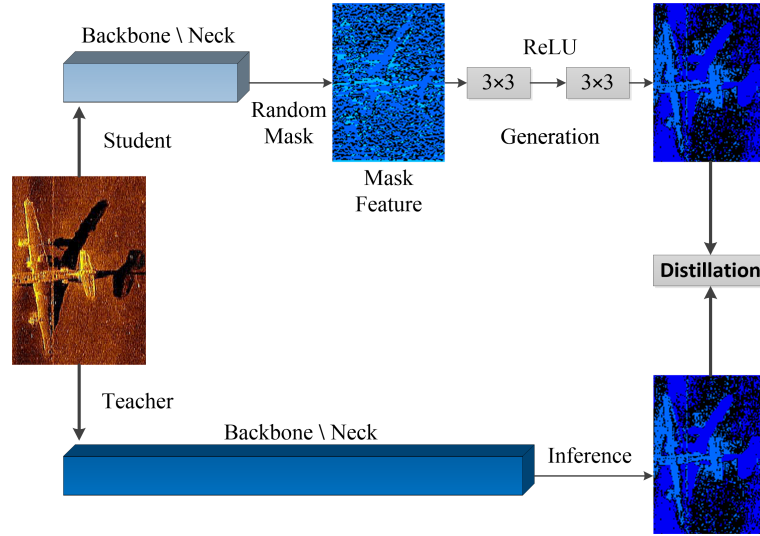


Figure 4. Diagram of the mask-generated distillation method. We use random masks to cover the feature maps of the student model and then utilize convolutional layers to forcefully generate the feature maps learned by the teacher model.

After obtaining the masked features, we use the mask of the l -th layer to cover the feature map of the student model's l -th layer. We then attempt to generate the teacher model's l -th layer feature map using the masked feature map. This process can be represented by Equations (13) and (14):

$$G(f_{cover}(S^l) \cdot M^l) \longrightarrow T^l \tag{13}$$

$$G(F) = W_{l2}(ReLU(W_{l1}(F))) \tag{14}$$

Here, F represents the feature map after masking, W_{l1} and W_{l2} are 3×3 convolutional layers, and ReLU is an activation layer between W_{l1} and W_{l2} . After computation, we obtain a feature map similar to the teacher model's l -th layer feature map. Based on this obtained feature map, we design the distillation loss function $L_{dis}(S, T)$ as Equation (15):

$$\mathcal{L}_{features}^{dis}(S, T) = \sum_{l=1}^L \sum_{k=1}^C \sum_{i=1}^H \sum_{j=1}^W \left(T_{k,i,j}^l - G\left(f_{cover}(S_{k,i,j}^l) \cdot M_{i,j}^l\right) \right)^2 \tag{15}$$

In this context, L represents the total number of layers to be distilled, C represents the number of channels in this layer, H and W denote the height and width of the feature maps, respectively. S represents the feature map generated by the student model, while T represents the feature map of the teacher model.

Combining the output distillation loss, the overall loss function of the distillation model is given by Equation (16):

$$L_{total} = \alpha \cdot \mathcal{L}_{features}^{dis}(S, T) + \beta \cdot \mathcal{L}_{logits}^{dis}(x) \tag{16}$$

4. Experiments and Analysis

To validate the effectiveness of the proposed lightweight method, this study conducted experiments on both the STCD dataset and a self-collected sidescan sonar dataset. Detailed analyses were performed on the detection accuracy and real-time performance of the model.

4.1. Experiment Platform

In this experiment, model training was conducted on a desktop computer running Ubuntu 20.04, with detailed specifications listed in Table 1. For testing, we used an embedded processor-Jetson AGX Orin, which can be installed inside the watertight compartment

of an AUV. The installation position on the AUV is illustrated in Figure 5, and detailed specifications are provided in Table 2.

Table 1. Configuration of desktop computer.

| Component | Specification |
|-----------------------------|-------------------------------|
| Operating System | Ubuntu 20.04 (64-bit) |
| Deep Learning Framework | Pytorch 1.11 |
| Programming Language | Python 3.9 |
| GPU Accelerated Environment | CUDA 11.3 |
| Graphics Card (GPU) | Nvidia GeForce RTX 3090 |
| Processor (CPU) | Platinum 8255C CPU @ 2.50 GHz |

Table 2. Configuration of embedded processor.

| Component | Specification |
|-----------------------------|--------------------------------|
| Operating System | Ubuntu 20.04 (64-bit) |
| Deep Learning Framework | Pytorch 1.12 |
| Programming Language | Python 3.9 |
| GPU Accelerated Environment | CUDA 11.3 |
| Graphics Card (GPU) | NVIDIA Ampere architecture GPU |
| Processor (CPU) | 8-core ARM Cortex-A78AE |



Figure 5. Schematic diagram of embedded processor installation location in AUV.

4.2. Dataset

The experiments in this paper utilized both the publicly available STCD dataset [53] and a self-collected dataset for testing. The STCD dataset is a public sidescan sonar dataset that includes three categories: ship, aircraft, and human, with a total of 497 images. In this study, the number of samples was increased to 692 through data augmentation and the addition of new data. The dataset was then split into training, validation, and test sets in a 7:1:2 ratio for experimentation. Our self-collected dataset was acquired during ocean

experiments using a dual-frequency sidescan sonar. This dataset comprises three categories: cylinder, cone, and non-target. After data augmentation using a diffusion model [54], it contains a total of 975 images. This dataset was also divided into training, validation, and test sets in a 7:1:2 ratio. The detailed sample numbers of the two datasets are shown in Table 3.

Table 3. The number of samples in the STCD dataset and OUR dataset.

| Category | STCD Dataset [53] | | | | OUR Dataset (256 × 256) [54] | | | | |
|----------|-------------------|-----|------|-------|------------------------------|-------|-----|------|-------|
| | Train | Val | Test | Total | Category | Train | Val | Test | Total |
| Ship | 252 | 37 | 72 | 361 | Cone | 205 | 29 | 59 | 293 |
| Aircraft | 126 | 18 | 36 | 180 | Cylinder | 223 | 31 | 64 | 318 |
| Human | 105 | 16 | 30 | 151 | Non-target | 255 | 36 | 73 | 364 |

4.3. Model Evaluation Metrics

To assess the effectiveness of the proposed compression method for object detection models [55,56], this paper will analyze the experimental results using the following technical metrics.

Params and Size: “Params” represent the total number of parameters in the model, and “Size” denotes the actual size of the model. These two parameters are directly related to the disk space required to deploy the model. For embedded devices, the limited storage space makes it impractical to deploy models that require a large amount of disk space. Therefore, these two parameters are valuable metrics for evaluating model compression techniques.

GFLOPs: “GFLOPs” represent the number of floating-point operations that can be performed per second, reflecting the model’s demand on hardware computational units. This metric is commonly used to assess the computational complexity or resource requirements of deep learning models. In model compression research, “GFLOPs” are often chosen as an evaluation metric. A higher “GFLOPs” value indicates a larger computational load and a greater need for computational resources.

Inter-time: “Inter-time” refers to the model’s inference time, which is the time required for the model to process input data and generate output results, excluding the time for non-maximum suppression. This metric most directly reflects the model’s inference speed. “Inter-time” is a crucial metric for evaluating the performance of compressed models, especially in real-time applications or scenarios requiring rapid responses.

mAP: mAP represents the overall performance of a model in detecting different categories of objects, comprehensively reflecting the model’s accuracy and recall capability in object detection tasks. It is a widely adopted evaluation metric in object detection tasks. “mAP@0.5” denotes the mean average precision at an IoU threshold of 0.5, while “mAP@0.5:0.95” represents the average mAP value over a range of IoU thresholds from 0.5 to 0.95.

4.4. Model Training and Compression

First, the YOLOv7 model was trained using the aforementioned datasets. After multiple trials, the model with the best performance on the validation set was selected as the baseline model for subsequent pruning and distillation experiments. The PR curves of the baseline model on the two datasets are shown in Figure 6, and the detailed parameters and performance metrics are provided in Table 4.

Table 4. Performance metrics of the baseline model.

| Dataset | Params (M) | Size (MB) | GFLOPs | Infer-Time (ms) | mAP@0.5 (%) | mAP@0.5:0.95 (%) |
|------------------|------------|-----------|--------|-----------------|-------------|------------------|
| STCD (640 × 640) | 36.5 | 71.4 | 103.2 | 9.7 | 92.84 | 71.01 |
| OUR (256 × 256) | 36.5 | 71.4 | 16.5 | 4.5 | 90.63 | 50.91 |

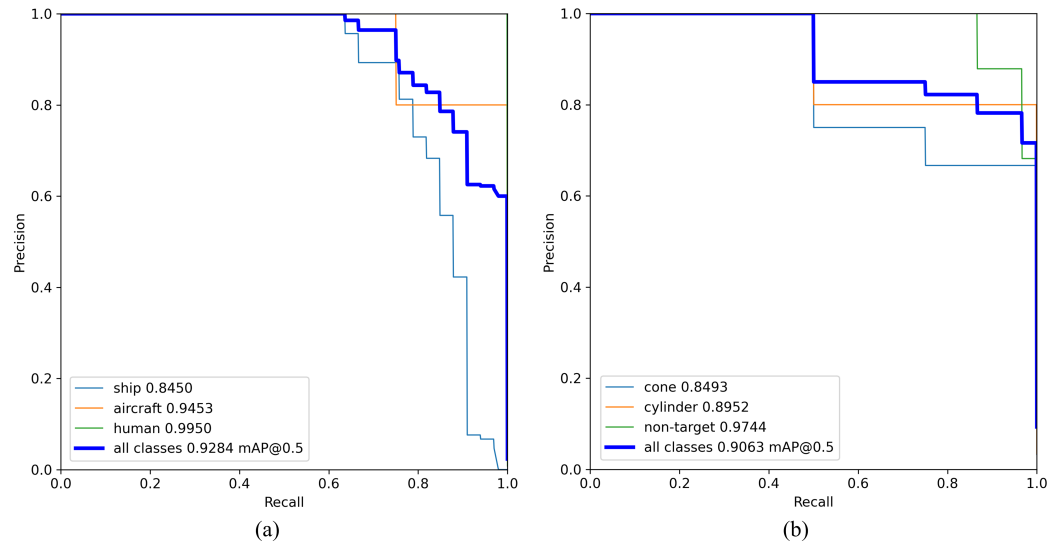


Figure 6. (a) The PR curves of the STCD dataset. (b) The PR curves of the OUR dataset.

Then, the obtained baseline model is pruned. In pruning, this paper refers to GFLOP values and sets compression ratios of 2, 4, and 6 times. After sparse training, the model undergoes automatic compression to the specified ratio or reaches the maximum compression times before automatic fine-tuning.

Finally, to improve the accuracy of the pruned model, the method proposed in this paper, which fuses output and features, is used to distill the pruned model. The performance of the pruned and distilled models is compared with that of the baseline model, as shown in Table 5.

Table 5. Comparison of performance metrics between the pruned model, distilled model, and baseline model.

| Dataset | Model | Params (M) | Size (MB) | GFLOPs | Infer-Time (ms) | mAP@0.5 (%) | mAP@0.5:0.95 (%) |
|------------------|-----------------|------------|-----------|--------|-----------------|-------------|------------------|
| STCD (640 × 640) | BaseLine | 36.5 | 71.4 | 103.2 | 9.7 | 92.84 | 71.01 |
| | Prune 2× | 26.0 | 50.8 | 50.5 | 8.6 | 92.11 | 69.84 |
| | KD 2× | 26.0 | 50.8 | 50.5 | 8.6 | 92.90 | 69.97 |
| | Prune 4× | 13.3 | 25.8 | 26.0 | 7.8 | 85.86 | 62.13 |
| | KD 4× | 13.3 | 25.8 | 26.0 | 7.8 | 91.85 | 69.95 |
| | Prune 6× | 7.8 | 15.3 | 17.0 | 6.8 | 77.77 | 49.12 |
| | KD 6× | 7.8 | 15.3 | 17.0 | 6.8 | 86.21 | 59.07 |
| | OUR (256 × 256) | BaseLine | 36.5 | 71.4 | 16.5 | 4.5 | 90.63 |
| Prune 2× | | 24.6 | 48.1 | 8.2 | 4.0 | 90.31 | 50.24 |
| KD 2× | | 24.6 | 48.1 | 8.2 | 4.0 | 90.61 | 50.88 |
| Prune 4× | | 13.0 | 25.1 | 4.1 | 3.4 | 86.14 | 47.20 |
| KD 4× | | 13.0 | 25.1 | 4.1 | 3.4 | 89.92 | 50.01 |
| Prune 6× | | 8.5 | 16.6 | 2.8 | 3.0 | 79.82 | 41.36 |
| KD 6× | | 8.5 | 16.6 | 2.8 | 3.0 | 84.57 | 45.26 |

Based on the results from the two datasets, it can be observed that under a compression ratio of 2 times, the detection accuracy of the pruned model is not significantly different from the baseline model. However, after applying the distillation method proposed in this paper, there is a slight improvement in the mAP metric. Under a compression ratio of 4 times, there is a noticeable decrease in detection accuracy for the pruned model. Using the distillation method proposed in this paper can improve the accuracy of the pruned model to a level close to that of the baseline model. Under a compression ratio of 6 times,

there is a significant decrease in detection accuracy for the pruned model, and even with the improvement from the distillation method, its accuracy still differs significantly from the baseline model, indicating that it may not meet practical detection requirements.

Based on the above analysis, in practical engineering applications, it is advisable to choose the model compressed by a factor of 4 times. This choice ensures that the model’s detection accuracy is maintained while maximizing compression to reduce model size and computational complexity, thus improving detection speed.

4.5. Model Performance Testing

To evaluate the real performance of the compressed models on embedded devices, we deployed both the baseline model and the distilled-enhanced compressed model on the Jetson AGX Orin for testing. The Jetson AGX Orin is a compact processor designed to be installed inside a 324-caliber AUV. In our testing experiments, we continued to use the two datasets mentioned earlier.

For the STCD dataset, the parameters and performance indicators of the compressed model compared to the baseline model are shown in Table 6. After distillation, the 4× compressed model reduced its parameter count by 23.2 M, model size by 45.6 MB, and computational load by 77.2 GFLOPs. The comparison of channel counts between the final compressed model and the baseline model is illustrated in Figure 7, demonstrating that our pruning method pruned most channels to varying degrees.

Table 6. Comparison of KD 4× model and baseline model on the STCD dataset.

| Model | Params (M) | Size (MB) | GFLOPs | Infer-Time (ms) | mAP@0.5 (%) | mAP@0.5:0.95 (%) |
|----------|------------|-----------|--------|-----------------|-------------|------------------|
| Baseline | 36.5 | 71.4 | 103.2 | 16.5 | 92.50 | 70.64 |
| KD 4× | 13.3 | 25.8 | 26.0 | 12.4 | 91.80 | 67.92 |

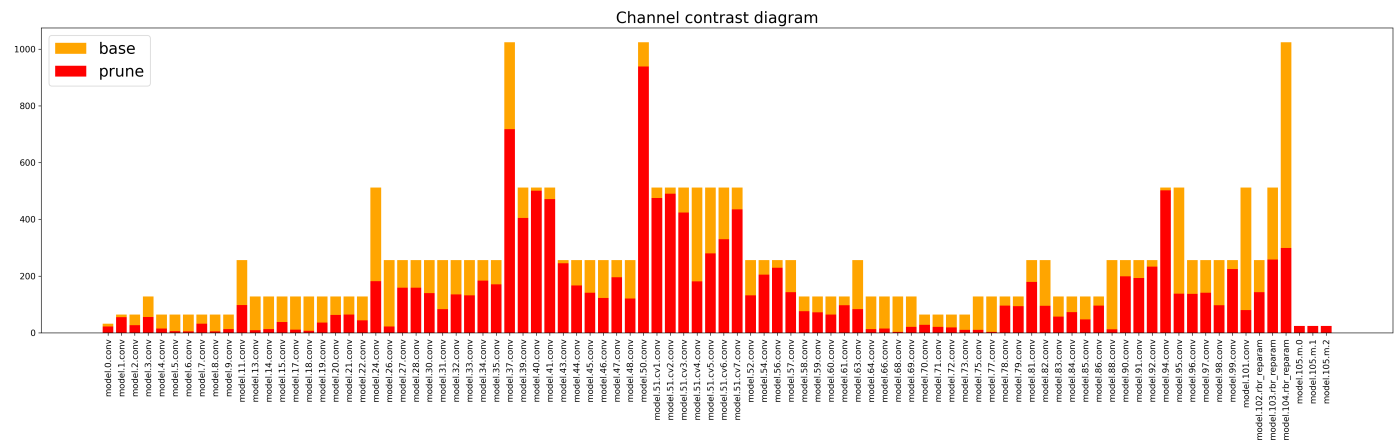


Figure 7. Comparison of channel numbers between the KD 4× model and the baseline model on the STCD dataset.

In terms of detection performance, the compressed model achieved an inference speedup of 4.1 ms compared to the baseline model. To ensure the rigor of the experiment, this inference time is the average result of 10 trials. The PR curve comparison between the compressed model and the baseline model on the test set is shown in Figure 8. The baseline model’s mAP@0.5 score is 92.50%, while the compressed model, after distillation enhancement, achieved a mAP@0.5 score of 91.80%, with only a slight difference of 0.7%. The detection results comparison between the compressed model and the baseline model is illustrated in Figure 9.

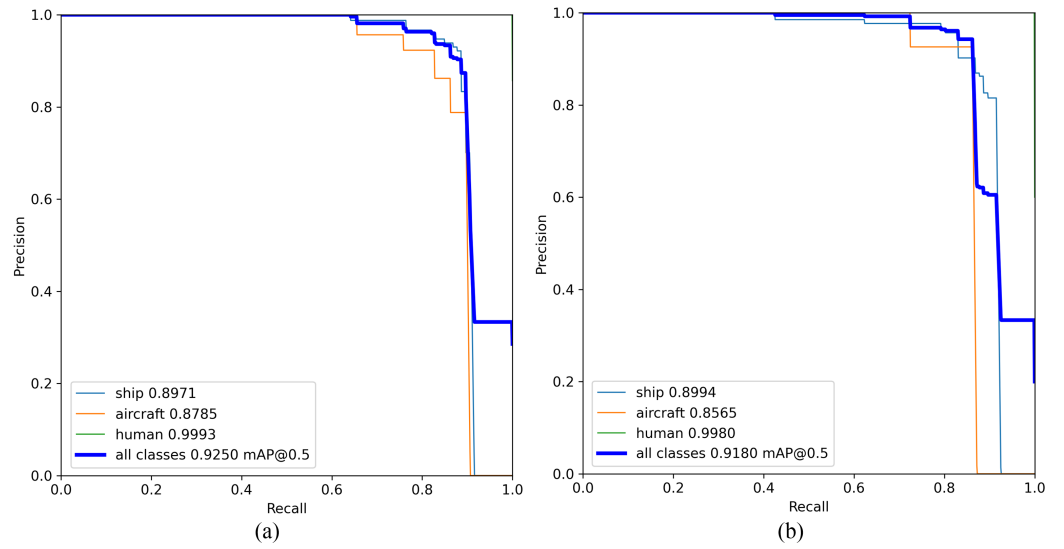


Figure 8. PR Curves of the KD 4× model and baseline model on the STCD dataset. (a) The KD 4× model. (b) The baseline model.

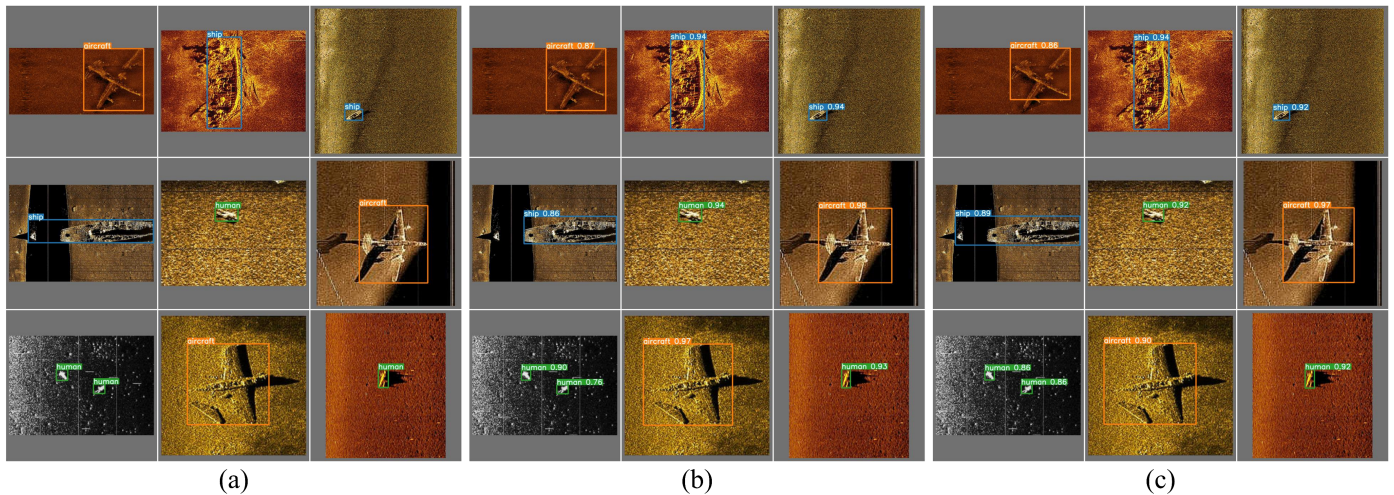


Figure 9. Detection performance of the KD 4× model and baseline model on the STCD dataset. (a) Ground-truth labels. (b) The KD 4× model. (c) The baseline model.

For the OUR dataset, the parameters and performance metrics of the compressed model compared to the baseline model are shown in Table 7.

Table 7. Comparison of KD 4× model and baseline model on the OUR dataset.

| Model | Params (M) | Size (MB) | GFLOPs | Infer-Time (ms) | mAP@0.5 (%) | mAP@0.5:0.95 (%) |
|----------|------------|-----------|--------|-----------------|-------------|------------------|
| Baseline | 36.5 | 71.4 | 16.5 | 8.1 | 89.93 | 52.76 |
| KD 4× | 13.0 | 25.1 | 4.1 | 5.8 | 89.32 | 48.79 |

In terms of model parameters, the KD 4× compressed model reduced the parameter count by 23.5 M, the model size by 46.3 MB, and the computational load by 12.5 GFLOPs. The comparison of the number of channels per layer between the compressed model and the baseline model is shown in Figure 10. Similarly, most of the channels underwent various degrees of pruning.

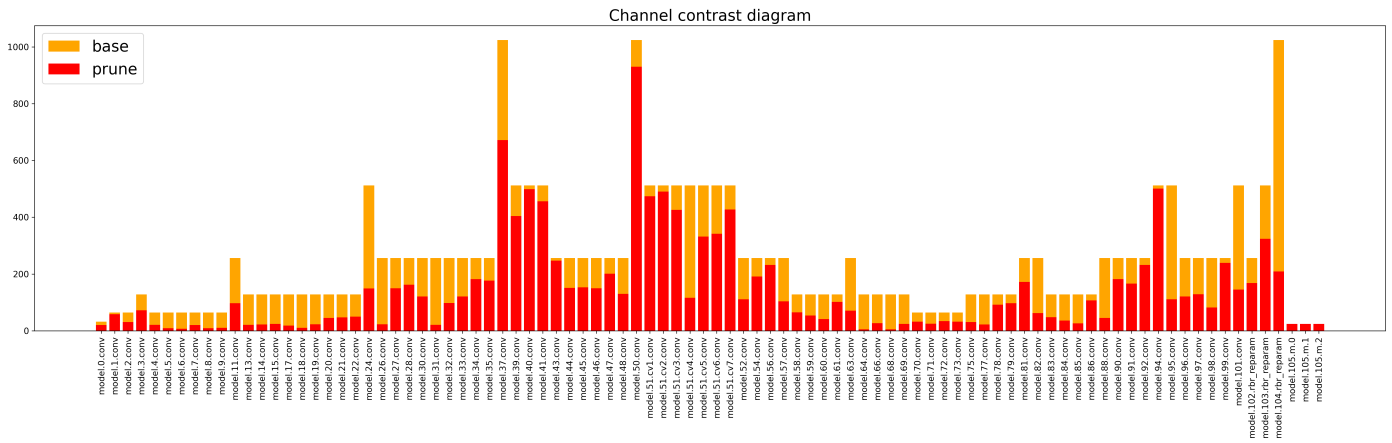


Figure 10. Comparison of channel numbers between the KD 4× model and the baseline model on the OUR dataset.

In terms of detection performance, the compressed model’s average inference speed improved by 2.3 ms compared to that of the baseline model. The PR curves for the compressed model and the baseline model on the test set are shown in Figure 11. The $mAP@0.5$ for the baseline model is 89.93%, while the $mAP@0.5$ for the compressed model, after distillation enhancement, is 89.32%, with only a difference of 0.61% from the baseline model. The comparison of the detection results between the compressed model and the baseline model is shown in Figure 12.

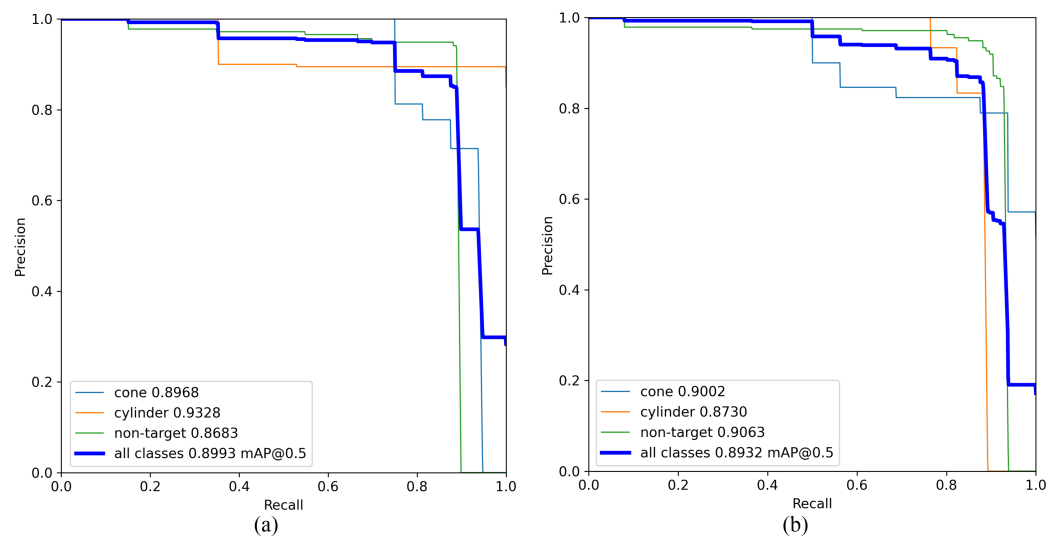


Figure 11. PR Curves of the KD 4× model and baseline model on the OUR dataset. (a) The KD 4× model. (b) The baseline model.

Analyzing the experimental results reveals that the proposed method of combining automated pruning with distillation can effectively compress dense detectors like YOLOv7 while maintaining nearly unchanged detection accuracy. Additionally, it becomes apparent that the pruning ratios set based on GFLOPs do not result in proportional reductions in other model parameters, which is influenced by both network and hardware structures. In practical engineering applications, it is essential to balance detection accuracy and model parameters by selecting a suitable compression ratio based on actual detection needs and the hardware platform.

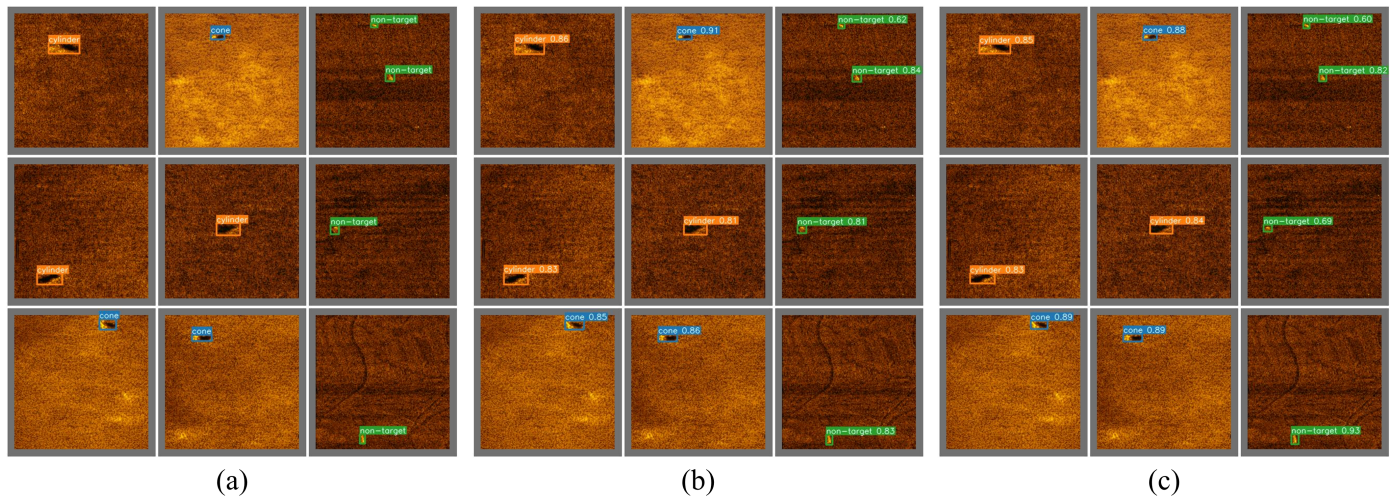


Figure 12. Detection performance of the KD 4× model and baseline model on the OUR dataset. (a) Ground-truth labels. (b) The KD 4× model. (c) The baseline model.

On the other hand, there are some considerations to note regarding the method presented in this paper. In terms of pruning, an automated pruning approach based on dependency graphs was used. This is a group-level pruning method that trims different dimensions across all parameter groups to achieve significant compression ratios. However, it may sometimes overly prune certain dimensions, such as reducing a dimension to 1, which can compromise final accuracy. In terms of distillation, the method employed combines output distillation and feature distillation, allowing the student network to learn more effectively from the teacher model. However, the distillation weight ratios need to be adjusted multiple times based on the dataset. For the dataset used in this study, the best results were obtained when the weights of output distillation and feature distillation together accounted for half of the total loss weight. In summary, excessive dimension pruning and improper distillation loss settings can lead to suboptimal results, but these issues can be resolved through multiple experiments.

5. Conclusions

This paper addresses the challenge of deploying large-scale detection models on edge devices by introducing a dependency graph-based pruning method, achieving efficient pruning on the YOLOv7 model. To counteract the significant accuracy drop caused by extensive pruning, we designed a hybrid distillation method combining output-based and feature-based distillation techniques to enhance the detection accuracy of the pruned model. Experimental validation demonstrates that our proposed method of combining pruning and distillation achieves a fourfold model compression with less than a 1% drop in accuracy. Finally, we deployed the compressed model on an embedded device within an AUV, reducing the model size and inference time while maintaining detection accuracy. These results confirm the effectiveness of our combined pruning and distillation approach.

Author Contributions: Conceptualization, C.C. and F.Z.; methodology, C.C., X.H. and C.W.; coding, C.C. and X.W.; designing the experiments, C.C. and X.H.; writing—original draft preparation, C.C.; writing—review and editing, C.C., F.Z. and W.L. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by the National Key R&D Program of China (2023YFC2808400).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: We would like to acknowledge the facilities and technical assistance provided by the Key Laboratory of Unmanned Underwater Transport Technology.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Fan, Z.; Xia, W.; Liu, X.; Li, H. Detection and segmentation of underwater objects from forward-looking sonar based on a modified Mask RCNN. *Signal Image Video Process.* **2021**, *15*, 1135–1143. [[CrossRef](#)]
- Zeng, L.; Sun, B.; Zhu, D. Underwater target detection based on Faster R-CNN and adversarial occlusion network. *Eng. Appl. Artif. Intell.* **2021**, *100*, 104190. [[CrossRef](#)]
- Yin, Q.; Yang, W.; Ran, M.; Wang, S. FD-SSD: An improved SSD object detection algorithm based on feature fusion and dilated convolution. *Signal Process. Image Commun.* **2021**, *98*, 116402. [[CrossRef](#)]
- Yundong, L.; Han, D.; Hongguang, L.; Zhang, X.; Zhang, B.; Zhifeng, X. Multi-block SSD based on small object detection for UAV railway scene surveillance. *Chin. J. Aeronaut.* **2020**, *33*, 1747–1755.
- Cheng, C.; Wang, C.; Yang, D.; Wen, X.; Liu, W.; Zhang, F. Underwater small target detection based on dynamic convolution and attention mechanism. *Front. Mar. Sci.* **2024**, *11*, 1348883. [[CrossRef](#)]
- Yu, Y.; Zhao, J.; Gong, Q.; Huang, C.; Zheng, G.; Ma, J. Real-time underwater maritime object detection in side-scan sonar images based on transformer-YOLOv5. *Remote Sens.* **2021**, *13*, 3555. [[CrossRef](#)]
- Aguirre-Castro, O.A.; García-Guerrero, E.E.; López-Bonilla, O.R.; Tlelo-Cuautle, E.; López-Mancilla, D.; Cárdenas-Valdez, J.R.; Olguín-Tiznado, J.E.; Inzunza-González, E. Evaluation of underwater image enhancement algorithms based on Retinex and its implementation on embedded systems. *Neurocomputing* **2022**, *494*, 148–159. [[CrossRef](#)]
- Jiang, M.; Li, R.; Liu, Q.; Shi, Y.; Tlelo-Cuautle, E. High speed long-term visual object tracking algorithm for real robot systems. *Neurocomputing* **2021**, *434*, 268–284. [[CrossRef](#)]
- Tang, Y.; Wang, L.; Jin, S.; Zhao, J.; Huang, C.; Yu, Y. AUV-based side-scan sonar real-time method for underwater-target detection. *J. Mar. Sci. Eng.* **2023**, *11*, 690. [[CrossRef](#)]
- Neves, G.; Ruiz, M.; Fontinele, J.; Oliveira, L. Rotated object detection with forward-looking sonar in underwater applications. *Expert Syst. Appl.* **2020**, *140*, 112870. [[CrossRef](#)]
- He, Y.; Xiao, L. Structured pruning for deep convolutional neural networks: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *46*, 2900–2919. [[CrossRef](#)] [[PubMed](#)]
- Zhou, Y.; Chen, S.; Wang, Y.; Huan, W. Review of research on lightweight convolutional neural networks. In Proceedings of the 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 12–14 June 2020; pp. 1713–1720.
- Ren, P.; Xiao, Y.; Chang, X.; Huang, P.Y.; Li, Z.; Chen, X.; Wang, X. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Comput. Surv.* **2021**, *54*, 1–34. [[CrossRef](#)]
- Liang, T.; Glossner, J.; Wang, L.; Shi, S.; Zhang, X. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing* **2021**, *461*, 370–403. [[CrossRef](#)]
- Gou, J.; Yu, B.; Maybank, S.J.; Tao, D. Knowledge distillation: A survey. *Int. J. Comput. Vis.* **2021**, *129*, 1789–1819. [[CrossRef](#)]
- Swaminathan, S.; Garg, D.; Kannan, R.; Andres, F. Sparse low rank factorization for deep neural network compression. *Neurocomputing* **2020**, *398*, 185–196. [[CrossRef](#)]
- Tian, M.; Li, X.; Kong, S.; Yu, J. Pruning-based YOLOv4 algorithm for underwater garbage detection. In Proceedings of the 2021 40th Chinese Control Conference (CCC), Shanghai, China, 26–28 July 2021; pp. 4008–4013.
- Szymak, P.; Piskur, P.; Naus, K. The effectiveness of using a pretrained deep learning neural networks for object classification in underwater video. *Remote Sens.* **2020**, *12*, 3020. [[CrossRef](#)]
- Ye, X.; Zhang, W.; Li, Y.; Luo, W. Mobilenetv3-YOLOv4-sonar: Object detection model based on lightweight network for forward-looking sonar image. In Proceedings of the OCEANS 2021: San Diego–Porto, San Diego, CA, USA, 20–23 September 2021; pp. 1–6.
- Yan, Z.; Shaochang, C. Lightweight improvement study of sonar image detection network. In Proceedings of the 2021 International Conference on Computer, Blockchain and Financial Development (CBFD), Nanjing, China, 23–25 April 2021; pp. 49–53.
- Hu, S.; Liu, T. Underwater rescue target detection based on acoustic images. *Sensors* **2024**, *24*, 1780. [[CrossRef](#)]
- Qin, K.S.; Liu, D.; Wang, F.; Zhou, J.; Yang, J.; Zhang, W. Improved YOLOv7 model for underwater sonar image object detection. *J. Vis. Commun. Image Represent.* **2024**, *100*, 104124. [[CrossRef](#)]
- LeCun, Y.; Denker, J.; Solla, S. Optimal brain damage. *Adv. Neural Inf. Process. Syst.* **1989**, *2*, 598–605.
- Han, S.; Pool, J.; Tran, J.; Dally, W. Learning both weights and connections for efficient neural network. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 1135–1143.
- Li, G.; Qian, C.; Jiang, C.; Lu, X.; Tang, K. Optimization based Layer-wise Magnitude-based Pruning for DNN Compression. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18), Stockholm, Sweden, 13–19 July 2018; pp. 2383–2389.
- Lee, J.; Park, S.; Mo, S.; Ahn, S.; Shin, J. Layer-adaptive sparsity for the magnitude-based pruning. *arXiv* **2020**, arXiv:2010.07611.

27. Carreira-Perpinán, M.A.; Idelbayev, Y. "learning-compression" algorithms for neural net pruning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8532–8541.
28. Kwon, S.J.; Lee, D.; Kim, B.; Kapoor, P.; Park, B.; Wei, G.Y. Structured compression by weight encryption for unstructured pruning and quantization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 1909–1918.
29. Molchanov, P.; Mallya, A.; Tyree, S.; Frosio, I.; Kautz, J. Importance estimation for neural network pruning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 11264–11272.
30. Wang, Z.; Li, C.; Wang, X. Convolutional neural network pruning with structural redundancy reduction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 14913–14922.
31. Ruan, X.; Liu, Y.; Li, B.; Yuan, C.; Hu, W. DPFPS: Dynamic and progressive filter pruning for compressing convolutional neural networks from scratch. *Proc. AAAI Conf. Artificial Intell.* **2021**, *35*, 2495–2503. [[CrossRef](#)]
32. Ding, X.; Hao, T.; Tan, J.; Liu, J.; Han, J.; Guo, Y.; Ding, G. Resrep: Lossless cnn pruning via decoupling remembering and forgetting. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 4510–4520.
33. Hou, Z.; Qin, M.; Sun, F.; Ma, X.; Yuan, K.; Xu, Y.; Chen, Y.K.; Jin, R.; Xie, Y.; Kung, S.Y. Chex: Channel exploration for cnn model compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 12287–12298.
34. Yang, T.J.; Chen, Y.H.; Sze, V. Designing energy-efficient convolutional neural networks using energy-aware pruning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 5687–5695.
35. Buciluă, C.; Caruana, R.; Niculescu-Mizil, A. Model compression. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, 20–23 August 2006; pp. 535–541.
36. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
37. Zhang, Y.; Xiang, T.; Hospedales, T.M.; Lu, H. Deep mutual learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4320–4328.
38. Kim, J.; Park, S.; Kwak, N. Paraphrasing complex network: Network compression via factor transfer. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 2760–2769.
39. Zhao, B.; Cui, Q.; Song, R.; Qiu, Y.; Liang, J. Decoupled knowledge distillation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 11953–11962.
40. Adriana, R.; Nicolas, B.; Ebrahimi, K.S.; Antoine, C.; Carlo, G.; Yoshua, B. Fitnets: Hints for thin deep nets. *Proc. ICLR* **2015**, *2*, 1.
41. Zagoruyko, S.; Komodakis, N. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv* **2016**, arXiv:1612.03928.
42. Huang, Z.; Wang, N. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv* **2017**, arXiv:1707.01219.
43. Ahn, S.; Hu, S.X.; Damianou, A.; Lawrence, N.D.; Dai, Z. Variational information distillation for knowledge transfer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 9163–9171.
44. Yang, J.; Martinez, B.; Bulat, A.; Tzimiropoulos, G. Knowledge distillation via adaptive instance normalization. *arXiv* **2020**, arXiv:2003.04289.
45. Yim, J.; Joo, D.; Bae, J.; Kim, J. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4133–4141.
46. Peng, B.; Jin, X.; Liu, J.; Li, D.; Wu, Y.; Liu, Y.; Zhou, S.; Zhang, Z. Correlation congruence for knowledge distillation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–November 2019; pp. 5007–5016.
47. Park, W.; Kim, D.; Lu, Y.; Cho, M. Relational knowledge distillation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 3967–3976.
48. Liu, Y.; Cao, J.; Li, B.; Yuan, C.; Hu, W.; Li, Y.; Duan, Y. Knowledge distillation via instance relationship graph. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 7096–7104.
49. Zhou, S.; Wang, Y.; Chen, D.; Chen, J.; Wang, X.; Wang, C.; Bu, J. Distilling holistic knowledge with graph neural networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 10387–10396.
50. Fang, G.; Ma, X.; Song, M.; Mi, M.B.; Wang, X. Depgraph: Towards any structural pruning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 16091–16101.
51. Yang, L.; Zhou, X.; Li, X.; Qiao, L.; Li, Z.; Yang, Z.; Wang, G.; Li, X. Bridging Cross-task Protocol Inconsistency for Distillation in Dense Object Detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 4–6 October 2023; pp. 17175–17184.
52. Yang, Z.; Li, Z.; Shao, M.; Shi, D.; Yuan, Z.; Yuan, C. Masked generative distillation. In *Computer Vision—ECCV 2022*; Springer: Cham, Switzerland, 2022; pp. 53–69.
53. Zhang, P.; Tang, J.; Zhong, H.; Ning, M.; Liu, D.; Wu, K. Self-trained target detection of radar and sonar images using automatic deep learning. *IEEE Trans. Geosci. Remote Sens.* **2021**, *60*, 1–14. [[CrossRef](#)]

54. Cheng, C.; Hou, X.; Wen, X.; Liu, W.; Zhang, F. Small-Sample Underwater Target Detection: A Joint Approach Utilizing Diffusion and YOLOv7 Model. *Remote Sens.* **2023**, *15*, 4772. [[CrossRef](#)]
55. Lyu, Z.; Yu, T.; Pan, F.; Zhang, Y.; Luo, J.; Zhang, D.; Chen, Y.; Zhang, B.; Li, G. A survey of model compression strategies for object detection. *Multimed. Tools Appl.* **2023**, *83*, 48165–48236. [[CrossRef](#)]
56. Zhou, Y.; Xia, L.; Zhao, J.; Yao, R.; Liu, B. Efficient convolutional neural networks and network compression methods for object detection: A survey. *Multimed. Tools Appl.* **2024**, *83*, 10167–10209. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.