Research on Rapid Detection of Underwater Targets Based on Global Differential Model Compression

Weishan Li , Yilin Li * , Ruixue Li , Haozhe Shen , Wenjun Li , Keqiang Yue

Posted Date: 22 August 2024

doi: 10.20944/preprints202408.1630.v1

Keywords: target detection; model compression; underwater embedded system; deep learning

 doi:10.20944/preprints202408.1630.v1

Article

Research on Rapid Detection of Underwater Targets Based on Global Differential Model Compression

Weishan Li [1], Yilin Li[1]*, Ruixue Li[1] ,Haozhe Shen[1], Wenjun Li[1] and Keqiang Yue[1]

School of Electronic Information,Hangzhou Dianzi Unvercity, Hangzhou, China

* Correspondence: ericlee@hdu.edu.cn; Tel.: +86-18681815615

Abstract: Large-scale deep learning models have produced significant results in underwater visual image target detection. However, deploying these models on underwater embedded devices poses a series of issues and challenges. Firstly, underwater devices have limited communication capabilities and carry relatively short supply of energy, resulting in constrained computing power for edge devices. Secondly, the large scale of deep neural networks creates a conflict in power demands with unmanned underwater equipment. Thirdly, insufficient underwater light and turbid water quality will lead to degradation problems in the images acquired by the device. In this paper, we build an offline system to complete the task of underwater scene target recognition. This paper introduces several innovations: 1) The new YOLO-TN model is developed by compressing the target recognition model based on YOLO-V5. The model use a model compression algorithm replace the YOLO-V5's backbone network with a lightweight network, then perform parameter pruning on the model detection head, while reduce the size of the input image. The model uses a Teacher-guided Neural Architecture Search method based on YOLO-V5, named as YOLO-TN. 2)A method for constructing and processing real underwater environment data sets is designed. The datasets are all captured using underwater unmanned vehicles in real underwater environments, addressing issues present in existing underwater datasets, such as unbalanced target volume and single-image environment. Experimental results demonstrate that the YOLO-TN model significantly reduces the model size while maintaining the original recognition accuracy. Furthermore, it achieves 28.6 FPS on embedded devices, which is 12 times that of the original model.

Keywords: target detection; model compression; underwater embedded system; deep learning

1. Introduction

With the development of artificial intelligence, humanity increasingly relies on highly automated unmanned underwater devices to assist in a variety of underwater operations in the ocean, such as cable defect detection, garbage cleaning and fishing.However, these technologies heavily depend on underwater target recognition systems. Only by locating their underwater mission targets, can unmanned underwater devices accomplish their designated tasks. Numerous traditional methods exist for underwater target recognition. Mainstream approaches include target detection based on underwater sonar images, and target recognition based on manually annotated features. But these methods have certain limitations. In recent years, as the computing power of high-performance hardware devices increases, the combination of deep learning and target recognition in underwater environments has gradually become mainstream [1]. Deep learning-based target recognition algorithms exhibit high detection accuracy, strong generalization capabilities, and can drastically improve the operational efficiency of underwater vehicles.

The previously proposed Regions with CNN features (R-CNN) [2] and Fast R-CNN [2,3] algorithms have addressed the problem of object classification. However, they suffer from high memory consumption, making it challenging to ensure real-time object detection. The You Only Look Once (YOLO) algorithm [4], as a single-stage object detection approach, adopts an end-to-end training methodology. It employs a convolutional neural network to predict, for each grid cell, whether the cell contains an object, the class of the object, and the positional information of the object. This enables

 doi:10.20944/preprints202408.1630.v1

efficient and real-time object detection. Afterwards, numerous improved algorithms in the YOLO series emerged, including PP-YOLOE [5], YOLO-V5 [6], among others. These advanced algorithms integrated various optimization methods, enhancing detection efficiency and effectiveness. Currently, YOLO-V5 can handle underwater blurry scenes, achieving a mAP of 57.5% in such scenarios.

The unparalleled inference performance of deep neural networks comes at the cost of vast memory consumption and high computational complexity. For instance, despite having only 8 layers, AlexNet [7] requires 61 million parameters, 731 million floating-point operations, and 233MB of memory. More complex Visual Geometry Group (VGG)-16 [8] reaches 138 million parameters, consuming around 553MB of memory. This shows that extensive deep neural networks lead to substantial GPU resource consumption. The immense computational resource requirements and memory overhead of large-scale deep neural networks pose challenges for deployment and application on platforms with constrained hardware resources, such as mobile devices and embedded systems. In underwater target recognition, electromagnetic waves experience significant attenuation in the underwater environment. Therefore, neural networks are challenged in real-time inference, because they cannot easily transmit data to servers for processing, as is common in terrestrial settings. Unmanned underwater devices equipped with underwater target systems are typically resource-constrained embedded devices. Directly deploying large-scale neural networks in such resource-limited environments significantly reduces inference speed, compromising real-time processing. In practical deployment scenarios, there is a need to ensure that the FPS (Frames Per Second) of the underwater target recognition model meets the real-time standard of FPS $\geq$ 25 [9]. In fact, the devices that deploy the model often suffer from insufficient computing power and resource constraints. The inference FPS on the CPU is 2.4. We analyzed the time consuming of the YOLO-V5s model through the Profiler tool of PyTorch. In the backbone network of YOLO-v5, most of the time is spent in the Cross Stage Partial (CSP)-1 module. By analyzing CSP1, it can be seen that the CSP1 Layer uses multi-channel convolution. Therefore, when the CSP1 Layer is used frequently or the number of CSP1 Layer channels is high, it will take up more Cache space and more CPU consumption. It is obvious that the YOLO-v5 network cannot meet the real-time requirements. Therefore, YOLO-v5 needs lightweighting operations to address the real-time challenges in deep underwater target recognition. It is obvious that the YOLO-v5 network cannot meet the real-time requirements. Therefore, YOLO-v5 needs lightweighting operations to address the real-time challenges in deep underwater target recognition.

Deep neural network model compression techniques can reduce the number of model parameters, save the model's memory and accelerate computations. Currently, there has been considerable progress in model compression-related work, with various methods such as network pruning, parameter quantization, lightweight module design and knowledge distillation continuously emerging.

Network pruning primarily accelerates inference speed by reducing model parameters and computational load. Many parameters contribute little to deviation reduction and network generalization in deep neural networks. After training, such parameters can be removed from the network. A simple pruning method was proposed to prune weights below a certain threshold and restore network accuracy by fine-tuning [10]. This process generates a sparse model after iterates. But this method only yields non-structured sparse models, the sparse model's runtime memory consumption and inference time do not significantly decrease compared with the baseline model. To address this, a method called Structured Sparsity Learning (SSL) [11] was Introduced to regularize filters, channels, filter shapes, and depth structures in Deep Neural Networks (DNN). This method enables DNNs to dynamically learn more compact structures without sacrificing accuracy. Later works demonstrated that convolutional kernels generating low-rank feature maps contribute little to network accuracy. An optimization function was defined to minimize the importance of removable convolutional kernels, and effectively remove some kernels [12]. Gao trained an independent neural network [13], which can predict the performance of a small network, and maximize the network's output to guide the pruning of the neural network. Parameter quantization [14] greatly reduces the number of computational operations by decreasing the number of bits used to represent weights and

 doi:10.20944/preprints202408.1630.v1

activation values, thereby reducing the size of trained deep neural networks. Quantization can be uniform [15,16] or non-uniform [17]. The main principle is to convert the weight and activation value from high precision to low precision through the mapping relationship between floating point and fixed point data.

The mentioned methods leverage the redundancy in parameters to reduce their quantity or decrease the number of computational operations. However, a drawback is that such methods often require a pre-trained large-scale DNN on which model pruning is performed to reduce its network size. This will lead to a decrease in the pre-trained model's accuracy, necessitating fine-tuning to restore model performance. On the other hand, designing lightweight modules is an emerging concept in model compression and acceleration. ShuffleNet [18] uses pointwise convolutions and channel shuffling to reduce the expensive cost of deep convolutional calculations and channel constraints. Fully Learnable Group Convolution (FLGC) [19] can be added to DNN models for acceleration. Knowledge distillation is also an emerging technology and research focus in the field of model compression [20]. The original knowledge distillation method used information from large models or model ensembles to guide the training of smaller models without dramatically reducing their accuracy [21]. Deriving an "attention map" from original feature maps express knowledge, which can guide the learning of the student network [22]. Later, using the activation boundary of hidden neurons for knowledge transfer was suggested [23], while a knowledge distillation method based on correlation consistency [24] was proposed. In terms of the analysis of knowledge distillation effects, researchers extensively analyzed the efficacy of knowledge distillation based on empirical evidence [25]. Larger models may not necessarily result in better knowledge distillation effects [26]. Similarly, The gap between large and small deep neural networks will reduce the effectiveness of knowledge transfer [27]. A globally differentiable deep neural architecture search method [28] introduces the knowledge of the teacher network into the network search, then automatically searches for an st-student network more sensitive to the teacher network's knowledge. The student network is able to replace the teacher network to perform the original task in some resource-constrained devices.

In this study, utilizing a lightweight network architecture search algorithm based on global gradients, we designed a lightweight underwater target recognition model constructed through Teacher-guided Neural Architecture Search search, named as YOLO-TN. This model was successfully deployed on embedded devices underwater, ensuring both high accuracy in model detection and resolution of computational conflicts. Furthermore, the construction and processing methods for a realistic underwater environment dataset proposed in this paper address challenges associated with imbalanced target quantities, a singular image environment, and unclear images in real underwater environments.

The main contributions of this paper are summarized as follows:

1) A lightweight underwater target recognition model, YOLO-TN is designed based on a search algorithm. Experimental results demonstrate that the YOLO-TN model achieves the mAP0.5 of 0.5425 after extreme compression with an input size of 416×416. This represents a reduction of less than 1% compared to YOLO-V5s. Moreover, the inference FPS reaches 28.8 on CPU, effectively realizing high accuracy and lightweight characteristics for the underwater target recognition model. This ensures the feasibility of offline deployment and real-time inference of the model.

2) Through the construction and processing methods of a realistic underwater environment dataset, issues such as imbalanced target quantities and a singular image environment in existing underwater datasets have been mitigated. Additionally, we analyzed challenges present in underwater datasets, including inadequate underwater illumination, image degradation, and blurring. For different real underwater environments, this paper proposed preprocessing techniques, including dark channel dehazing, underwater image color restoration, and automatic color balance algorithms. These methods effectively enhance the quality of underwater datasets, improving the model's generalization.

3) Deploying the YOLO-TN model to the Jetson TX2 embedded platform using the MNN inference engine, a real-time underwater target offline recognition system is established. Test results indicate

 doi:10.20944/preprints202408.1630.v1

that the FPS of the YOLO-TN network ensures real-time performance. The pruned YOLO-TN achieves FPS of 28.6 and 20.4 at input sizes of 320×320 and 416×416, respectively.

The remainder of the paper is organized as follows. Section 2 provides the method of building the model. Section 3 introduces the model's performance on public datasets. In Section 4, the model is deployed on-device and evaluated in real experiments. And Section 5 concludes this paper.4

2. Model Construction

In this work, we employ a globally differentiable deep neural architecture search method, simultaneously introducing the knowledge of the teacher network into the search process. It automatically searches a student network capable of learning more knowledge from the teacher network. This student network can replace the teacher network in resource-constrained devices, maintaining performance close to the original. The Differentiable ARchiTecture Search (DARTS) method [29] is utilized for architecture search. In contrast to inefficient black-box searches, gradient-based optimization enables DARTS to achieve performance levels comparable to existing neural architecture search technologies using fewer computational resources. This lightweight network architecture search constructs a compact model with low storage requirements, low computational complexity, and excellent network performance. Parameter pruning can further remove redundancies from a trained model, thus, achieving a small-sized, fast-inference, and high-accuracy underwater target offline recognition model. This improved lightweight underwater target recognition model is named "YOLO-TN." In the search process using the globally differentiable deep network architecture search compression algorithm [28], the search space is defined as 3×3 depth-wise separable convolution, 5×5 depth-wise separable convolution, 3×3 dilated convolution, 5×5 dilated convolution, 3×3 max pooling, 3×3 average pooling, and skip connections. This determines the operations between different nodes in the basic computing unit. To better recognize small targets, operations in the search space minimize the size and stride of convolution kernels, avoiding the omission of feature extraction for some small targets. Therefore, all operation strides are set to 1 in the search space. The final architecture consists of operations present in the search space.

2.1. Loss Function

The total loss function for the network is represented as:

$$L_{YOLO} = \sum_{i}^{N} \left(\lambda_1 L_{CIoU} + \lambda_2 L_{conf} + \lambda_3 L_{cls} \right) \tag{1}$$

where L_{CIoU},L_{conf},L_{cls}denote the bounding box loss, confidence loss, and classification loss, respectively.λ_1,λ_2,$\lambda_3$are the weights for the three losses. To prevent "learning" from predicting the background region, in YOLO's distillation, the distillation loss is formulated as an objective scale function. The student network learns bounding box coordinates and target class probabilities only when the teacher network predicts a high confidence for an object. The guided confidence loss for distillation is represented as:

$$L_{conf}^{Comb}(o_i^{gt}, \widehat{o_i}, o_i^T) = \underbrace{L_{conf}(o_i^{gt}, \widehat{o_i})}_{\text{Distection loss}} + \underbrace{\lambda_D \cdot L_{conf}(o_i^T, \widehat{o_i})}_{\text{Distillation loss}} \tag{2}$$

where $\widehat{o_i}$ is the predicted confidence of the student network, o_i^{gt} is the true confidence label, o_i^T is the predicted confidence of the teacher network. The classification loss for the student network is:

$$L_{cls}^{Comb}\left(p_i^{gt}, \widehat{p_i}, p_i^T, \widehat{o_i^T}\right) = L_{cls}\left(p_i^{gt}, \widehat{p_i}\right) + \widehat{o_i^T} \cdot \lambda_D \cdot L_{cls}\left(p_i^T, \widehat{p_i}\right) \tag{3}$$

 doi:10.20944/preprints202408.1630.v1

where $\widehat{p}_i$ is the predicted probability of the student network containing the target in the predicted box, p_i^{gt} is the true value of the target category, p_i^T is the predicted probability of the teacher network containing the target in the predicted box. Similarly, the bounding box loss for the student network is represented as:

$$L_{CIoU}^{Comb}\left(b^{gt}, \widehat{b}_i, b_i^T, \widehat{o}_i^T\right) = L_{CIoU}\left(b_i^{gt}, \widehat{b}_i\right) + \widehat{o}_i^T \cdot \lambda_D \cdot L_{CIoU}\left(b_i^T, \widehat{b}_i\right) \tag{4}$$

where $\widehat{b}_i$ is the predicted box coordinates of the student network, b_i^{gt} is the true bounding box coordinates, b_i^T is the predicted box coordinates of the teacher network. Overall, in the improved loss function, knowledge of the teacher model in detecting the target is added, and the predictions for the background units are discarded. The total loss function during training is represented as:

$$L_{YOLOKD} = L_{conf}^{Comb}\left(o_i^{gt}, \widehat{o}_i, o_i^T\right) + L_{cls}^{Comb}\left(p_i^{gt}, \widehat{p}_i, p_i^T, \widehat{o_i^T}\right) + L_{CIoU}^{Comb}\left(b^{gt}, \widehat{b}_i, b_i^T, \widehat{o}_i^T\right). \tag{5}$$

During the dual-layer optimization process, $L_{YOLO-KD}$ is used to optimize the network parameters w.

In optimizing the architectureα and simultaneously searching for the backbone network of the underwater target recognition model, the sigmoid function is used instead of SoftMax, to avoid the phenomenon of skip connection enrichment during the search phase. To make the distinction between different operations more pronounced and to enhance the binary nature of the architecture weights, a 0-1 loss is introduced to ensure that $\sigma(\alpha_i)$ are closer to 0 or 1. Using the weight coefficient w_{0-1} to control the intensity of the 0-1 loss, the total loss for searching architecture α is:

$$L_{\alpha-total} = \mathcal{L}_{val}(w^\flat(\alpha), \alpha) + w_{0-1}L_{0-1} \tag{6}$$

where L_{0-1} is the 0-1 loss.

2.2. Construction of the Student Model

For a deep neural network performing classification tasks, we stack a super net at the cell level. This super net serves as the search space to find the optimal cell, thus obtaining the most suitable super net. We similarly stack a super net shaped like YOLO to search for an architecture α suitable for underwater datasets in the context of underwater object recognition networks.

In constructing the super net for object recognition tasks, to obtain different receptive fields and balance the handling of small objects in underwater recognition, downsampling is performed at appropriate computational nodes, similar to the YOLO original framework. In YOLO-v5, a 32x downsampling of the feature map is obtained through five convolution operations. Therefore, for the construction of the super net used in object recognition, we also use 32x downsampling. Achieving a 32x downsampling requires three ordinary convolutional layers and two basic computational units. The structure is shown in Figure 1

 doi:10.20944/preprints202408.1630.v1

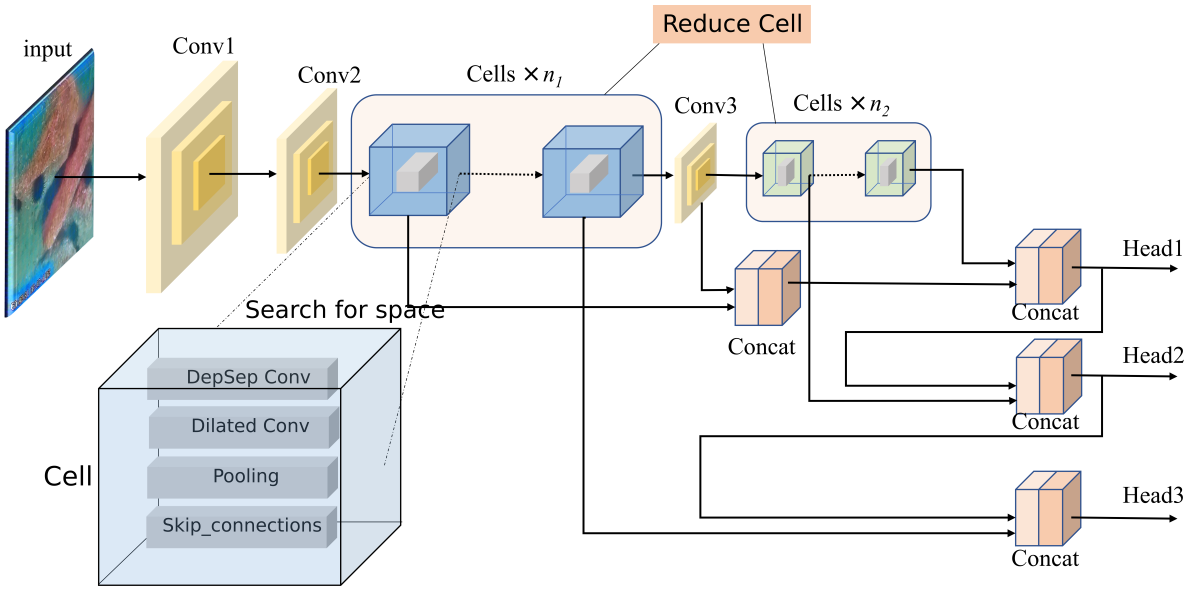


Figure 1. Schematic Diagram of the Search Network Structure. This network framework uses three normal convolutional layers and two basic computing units to achieve 32x downsampling, while sets up Reduce computing units at 1/3 and 2/3 depth of the network, stacked as the backbone network.

The basic computational unit is divided into two types: Normal and Reduce. Similarly, at depths of 1/3 and 2/3 of the network, Reduce computational units are set, and the rest are Normal computational units. Stacking these basic computational units, the backbone network of the underwater target recognition network is formed. In this paper, the lightweight backbone network obtained through neural architecture search is named the "TN" (Teacher guide Neural Architecture Search) module. After obtaining the lightweight backbone network TN, similar to YOLO-v5, detection heads are placed at downsampling positions of 32x, 16x, and 8x in the entire object recognition network's feature maps to get the model structure of YOLO-TN.

2.3. Parameter Pruning

To further compress the underwater target recognition model and improve inference frames per second (FPS), the pruning algorithm is employed to trim the detection head of YOLO-TN drastically. This paper introduces a straightforward yet effective network pruning method. L1 regularization is applied to the scaling factors in the Batch Normalization (BN) layer, pushing the values of BN layer scale factors toward zero, thereby identifying irrelevant channels[25]. After pruning, any accuracy loss induced by pruning can be mitigated through retraining.

2.4. Evaluation Criteria

This study aims to deploy the model to resource-constrained embedded systems, necessitating additional metrics. Floating Point Operations per Second (FLOPs) represents the floating-point operations required by the network, indicating the computational load. A lower FLOPs value for image processing on a particular device implies that it can process more images in the same time frame. Frames Per Second (FPS) is a common concept in object recognition and image processing. In the domain of object recognition, a higher FPS implies that more images can be detected per second, ensuring better real-time performance.

The number of parameters represents the sum of all parameters that need to be trained during model training. It determines the model's size, affecting the memory or GPU memory it occupies. Generally, a smaller number of model parameters result in fewer FLOPs, less storage space required to save the model, and lower hardware requirements.

 doi:10.20944/preprints202408.1630.v1

In object recognition tasks, evaluating a model's performance is challenging as it often involves classifying multi-label images. Mean Average Precision (mAP) is a widely used metric to assess model performance in object recognition tasks. mAP represents the average value of Average Precision (AP) for each category, where AP is the Average Precision that evaluates the detection performance for a specific class. It is calculated as the area under the Precision-Recall curve. By computing the AP for each class, summing them, and then taking the average, the mAP value can be obtained. mAP can be further categorized based on different Intersection over Union (IoU) thresholds, such as mAP0.5 (IOU threshold is 0.5) to mAP0.95.

3. Model Validation

This study conducts experiments using a multi-GPU training and single-GPU/CPU inference mode. The hardware setup is as follows:

Server Configuration:

- CPU: Intel(R) Xeon(R) Silver 4110
- Memory: 36GB DDR4
- GPU: Nvidia 2080TI

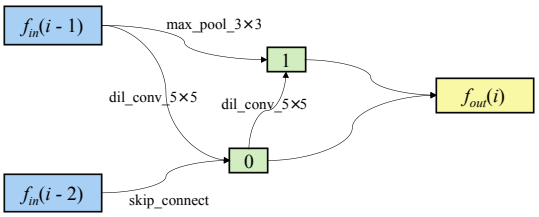
Software Configuration:

- Ubuntu 18.04
- PyTorch 1.9.0
- CUDA 11.2

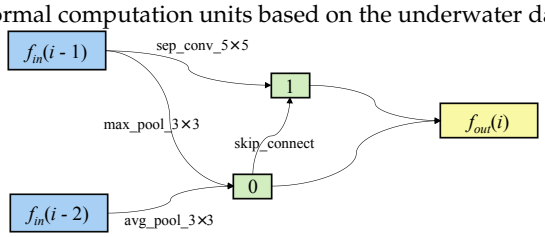
The experiment begins by using the architecture obtained from the super net as the basic building block. Different experiments are conducted by stacking various parameters to validate the model performance and inference speed of the YOLO-TN model under different parameter settings. The dataset used is the fish-trash dataset officially included in the YOLO collection. This dataset contains 1823 clear images of plastic waste and fish.

The experimental results confirm that the architecture obtained using the network architecture search method in this paper exhibits superior performance on the given dataset compared to conventional search methods and is more lightweight. After comparative search experiments, the optimal search architecture α suitable for a real underwater dataset is obtained. The Normal and Reduce computation units constructed using α are shown in Figure 2.

 doi:10.20944/preprints202408.1630.v1



(a) Normal computation units based on the underwater dataset.



(b) Reduce computation units based on the underwater dataset.

Figure 2. Each computational unit is primarily composed of 5 nodes, and the output is the cascaded output of all intermediate nodes within the computational unit. The first and second input nodes of the i-th computational unit are set as the outputs of the (i-2)-th and (i-1)-th computational units, respectively, and a 1x1 convolution may be inserted as needed. The first intermediate node is obtained by linearly transforming the two input nodes, adding their results, and then applying the tanh activation function. Other configurations are similar to what ENAS's cell does, enabling batch normalization in each node to prevent gradient explosions during architecture search.

After determining the structure of the basic computation unit used to construct the offline underwater target recognition algorithm, further experiments are conducted by changing the input channels and the number of basic units for different cells. The goal is to obtain an underwater target recognition network that is lightweight and has excellent performance. The experimental results are shown in Table I.

Table 1. Experimental Results for Different Initial Channels and Basic Unit Numbers.

Model	Cell number	initial channel	mAP0.5 (Undistilled/Distilled)	Parameter (M)	FPS (GPU/CPU)	FLOPs (G)
YOLO-TN(a)	10	8	0.5038/0.5205	2.8704	112.7/9.8	7.8
YOLO-TN(b)	10	16	0.5312/0.5441	3.0516	109.4/7.7	9.1
YOLO-TN(c)	7	16	0.5326/0.5437	1.2083	134.5/8.9	3.9
YOLO-TN(d)	5	16	0.5355/0.5471	0.9481	162.7/10.2	3.5
YOLO-TN(e)	4	16	0.5384/0.5592	0.8401	176.3/14.1	3.3
YOLO-v5s	-	-	0.5495/-	7.2	178.9/8.3	16.5

When comparing YOLO-TN(a) and YOLO-TN(b), increasing the initial channels significantly enhances mAP. However, it also leads to a decrease in FPS and an increase in FLOPs, which is not favorable for real-time inference on embedded devices. Similarly, increasing the number of cell units may result in the failure to extract features for small targets, while too few units weaken the effectiveness of the super network. Balancing model effectiveness and real-time inference, we choose YOLO-TN(e) with only 4 basic computational units (already sufficiently small) and an initial channel of 16 for further compression. The complete structure of YOLO-TN(e) is illustrated in Figure 3.

 doi:10.20944/preprints202408.1630.v1

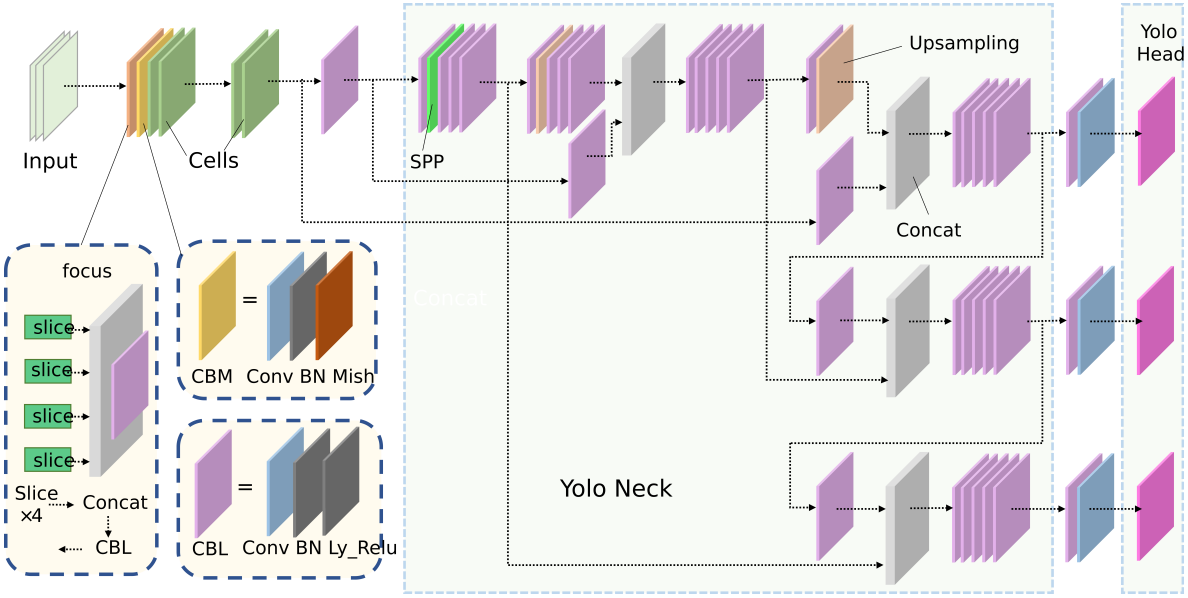


Figure 3. YOLO-TN(e) Complete Structure. The new architecture employs depthwise separable convolution, which divides a standard convolution into depthwise convolution and pointwise convolution, leading to a reduction in the number of parameters and computational workload.

Later, the network structure undergoes parameter pruning for the final extreme compression. Pruning rates ranging from 0.4 to 0.8 with a step size of 0.1 are set. This paper explore the effects of channel pruning based on BN layer coefficients and convolutional kernel pruning based on L2 norm (referred to as prune-A and prune-B). Through analysis of experimental results, it is observed that models pruned using the prune-A strategy generally exhibit higher accuracy than the latter under the same pruning rate. Moreover, at a pruning rate of 0.5, it achieves the optimal mAp0.5 value in the experiment.

 doi:10.20944/preprints202408.1630.v1

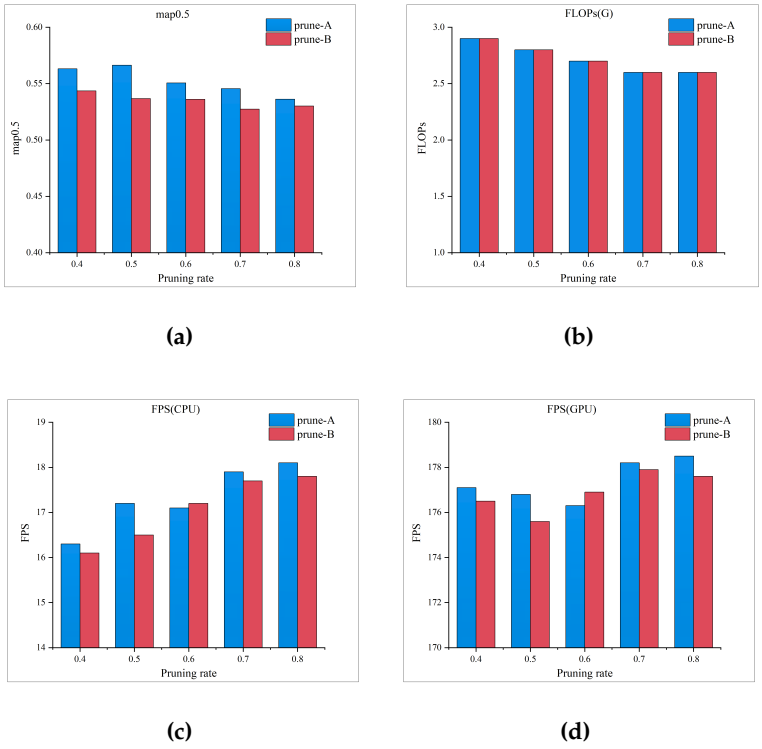


Figure 4. Model Performance Comparison with Different Pruning Strategies and Rates

As observed in Figure 4, the lightweight network proposed in this paper, YOLO-TN, achieves a smaller parameter count and faster inference speed compared to the benchmark models. Considering both mAp0.5 value and inference speed, the YOLO-TN model pruned using strategy prune-A at a pruning rate of 0.5 stands out as the optimal choice, striking a balance between model recognition accuracy and real-time performance.

Finally, we apply the complete network structure to the dataset, and the training results are compared with YOLO-v5s, as illustrated in Figure 5.

 doi:10.20944/preprints202408.1630.v1

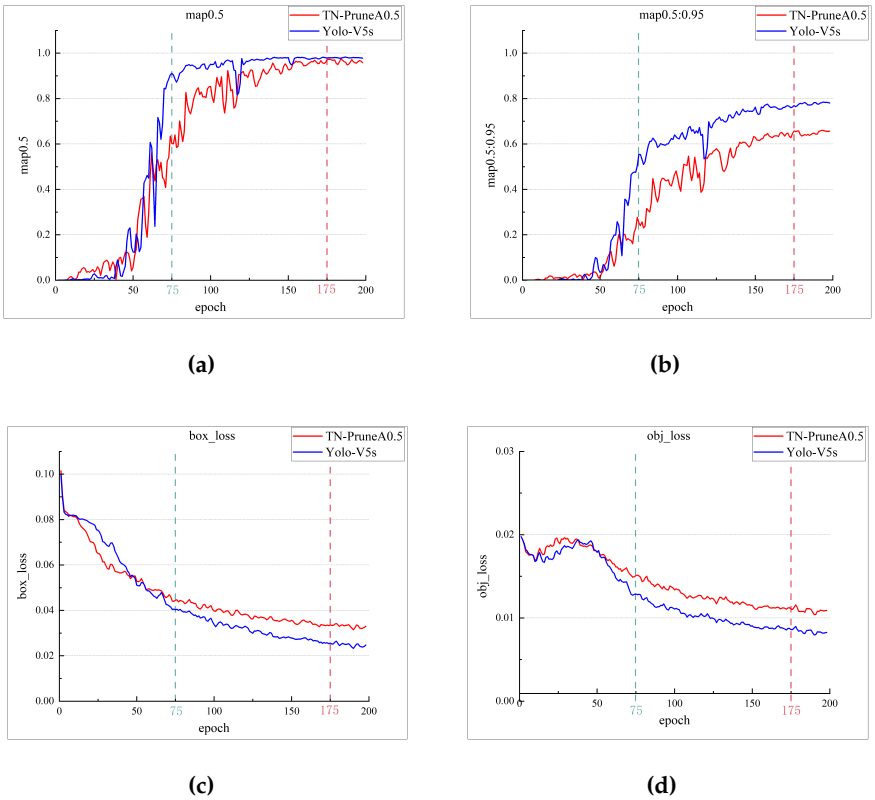


Figure 5. Training mAp and Loss Curves Using the Fish-Trash Dataset

In Figure 5, the training data for YOLO-TN-pruneA-0.5 is compared with that of YOLO-v5s using the fish-trash dataset. The mAp0.5 curve indicates that the lightweight network rapidly achieves a model with superior performance. The loss curve demonstrates that the training loss, guided by the teacher network, decreases more rapidly and converges earlier. This is attributed to the lightweight nature of the model, facilitated by the depth-wise separable convolutional structure, allowing for a swift learning of sample features and parameter updates.

At an input image size of 640×640, YOLO-TN's FPS outperforms the listed models in this paper. Nevertheless, for devices with limited resources, ensuring real-time predictions remains challenging. Therefore, training is performed by reducing the input image size, and the model's performance is tested with reduced input dimensions of 416×416 and 320×320. The results are presented in Table II.

Table 2. Model Experiment Results with Different Input Sizes

Model	Input Size	mAP0.5	FPS(GPU/CPU)	FLOPs(G)
YOLO-TN-640	640×640	0.5592	176.8/17.2	3.3
YOLO-TN-416	416×416	0.5425	176.9/28.8	2.8
YOLO-TN-320	320×320	0.5101	177.6/38.4	2.8

4. Engineering Experiment

Underwater offline recognition models are often deployed on a variety of embedded platforms, enabling underwater unmanned devices to autonomously complete underwater tasks in offline mode. To achieve lightweight deployment and accelerate computation speed, this paper based on the MNN inference engine framework, deploys the obtained YOLO-TN model on the NVIDIA Jetson TX2 development platform for embedded endpoint deployment. This further verifies the model's

 doi:10.20944/preprints202408.1630.v1

performance in resource-constrained embedded platforms, realizing a real-time underwater offline recognition system.

4.1. Evaluation Criteria

To replicate a real underwater environment, the dataset used in this study was mostly collected from the waters of Qiandao Lake in Hangzhou, Zhejiang Province. Another small portion was collected from a scenic fish pond in the northern district of Qiandao Lake City. Data was collected using the Youcan BW SpacePro underwater drone equipped with a SONY 4k HD camera at a depth of 0.5-30 meters. Images were saved in video format with a resolution of 1920×1080, and frames were extracted at 1-second intervals, resulting in a total of 5964 valid images. As shown in Figure 6. Considering the issue of target size and quantity in training image samples, the network's receptive field and layer numbers were studied. However, the real underwater environment introduces image degradation due to light attenuation caused by the scattering and absorption of water, leading to issues like illumination decay, detail blurring, and low contrast. To address these problems, especially in underwater object recognition tasks, image restoration processing is crucial to prevent potential issues like false positives.

4.2. Data Preprocessing

This paper utilizes the dark channel dehazing algorithm to address issues like image degradation, blurring, and low visibility caused by turbid underwater images. The underwater color restoration algorithm, which considers visible light attenuation, is employed to tackle image degradation due to insufficient underwater lighting, restoring its original color. Additionally, an automatic color balance algorithm is used to process images with color distortion in deep underwater conditions. After processing, the color histogram ratio of the image is close to that of the ground environment. Image processing algorithms, as preprocessing steps, are generally applied during the training phase to enhance the dataset and further improve model generalization.

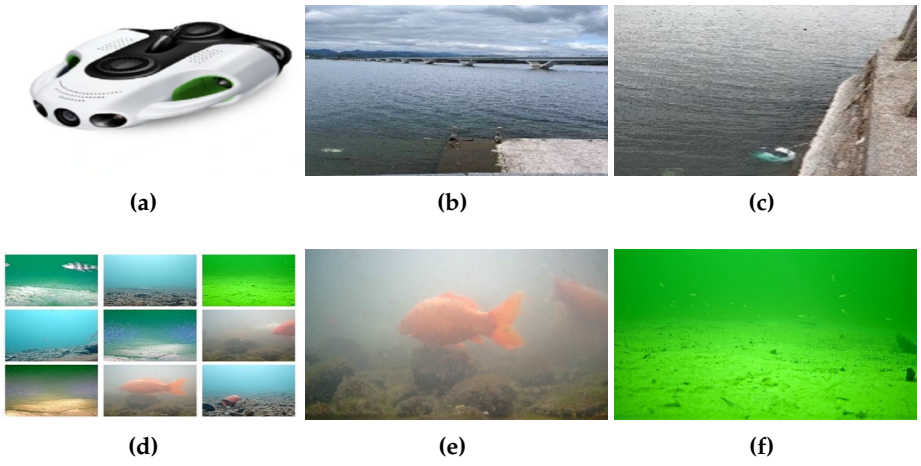


Figure 6. Model Performance Comparison with Different Pruning Strategies and Rates. (a) is the device used for shooting. (b) is the field photo of Qiandao Lake. (c) is a field photo of a park. (d) is the sum of the collected data. (e) and (f) are the two images in the data set.

 doi:10.20944/preprints202408.1630.v1

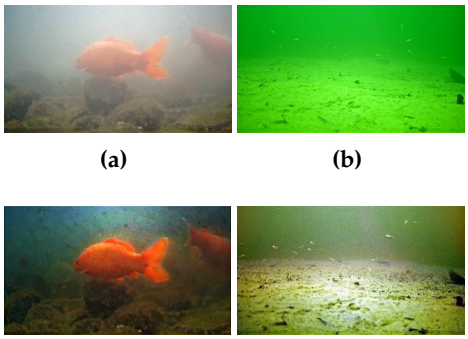


Figure 7. Dark Channel Dehazing Algorithm Processing Blurred Image And Automatic Color Equalization Algorithm Processing Underwater Deep Color Distortion Image. (c) is the image processed by the dark channel defogging algorithm in (a). (d) is the image processed by the automatic color balance algorithm in (b).

4.3. Model Application

The processed dataset is used for training in the network, and the training results of YOLO-TN-Prune-A0.5 and YOLO-V5s are shown in Figure 8.

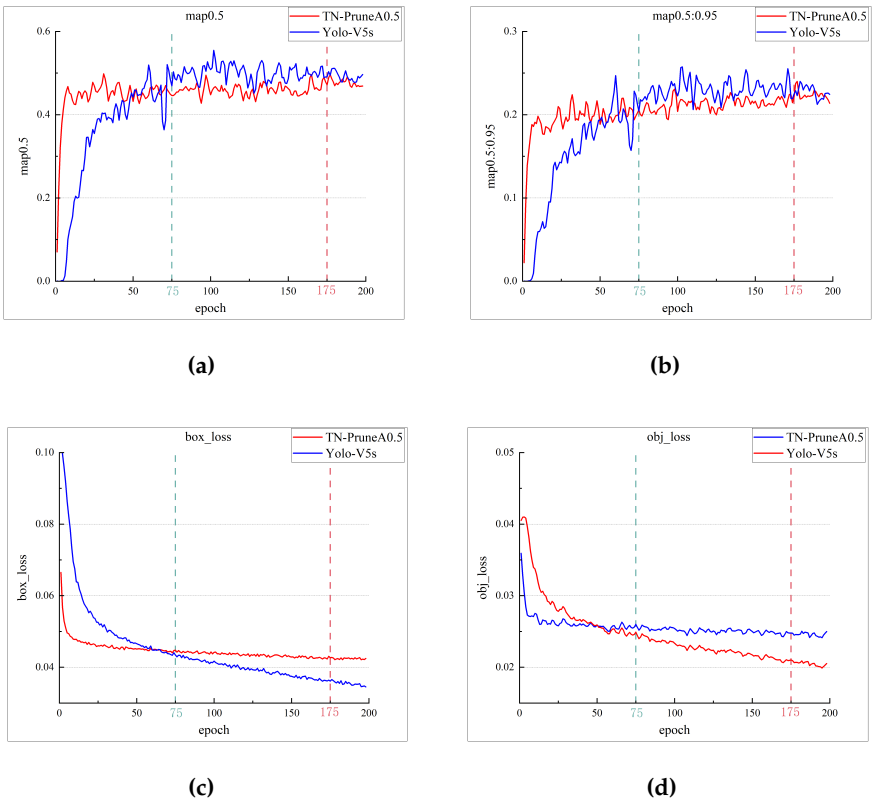


Figure 8. Training mAp and Loss Curves Using Real Underwater Dataset

In this study, YOLO-v5s, unpruned YOLO-TN, and pruned YOLO-TN models with three input sizes are selected as experimental models. After converting the experimental models uniformly from .pth format to .mnn format, they are loaded into the MNN inference framework for inference experiments. The experimental results are shown in Table III..

 doi:10.20944/preprints202408.1630.v1

Table 3. MNN Inference Framework Inference FPS on TX2

Model	Input Size	FPS(CPU)
Pruned YOLO-TN-640	640×640	10.8
Pruned YOLO-TN-416	416×416	20.4
Pruned YOLO-TN-320	320×320	28.6
Unpruned YOLO-TN	640×640	8.9
YOLO-v5s	640×640	2.4

The YOLO-TN with an input size of 320×320 achieves real-time performance in inference and simultaneously ensures good recognition results. This suggests that the model can be deployed on Jetson TX2 to complete a real-time underwater object recognition system. Figure 9 displays the recognition results of different models in different scenarios. It can be observed that, at an input size of 320×320, the pruned YOLO-TN recognizes a similar number of objects with accuracy comparable to YOLO-v5s. In terms of inference speed, its FPS is nearly 12 times that of YOLO-v5s.

5. Conclusions

This paper addresses the challenge posed by the high computational demands of the YOLO-v5 model, enabling the YOLO model to perform efficiently on offline embedded devices. We propose an enhanced lightweight underwater target network and establish a real-time underwater target offline recognition system. Firstly, we construct a feature extraction network with a small parameter count based on a specialized network search algorithm, replacing the backbone of YOLO-v5. Simultaneously, applying various compression methods and combining parameter pruning, we ultimately obtain the lightweight target detection model, YOLO-TN. Furthermore, this study collects a dataset under real underwater conditions, addressing challenges such as the scarcity of underwater target datasets, single-image targets, overly idealized collection environments, and excessively static targets. Multiple image processing algorithms are employed to enhance the dataset. Building upon this, we establish a real-time underwater target offline recognition system. The system, based on the high-performance inference framework MNN, is deployed on the NVIDIA Jetson TX2 embedded platform, capable of achieving real-time inference for underwater targets. Utilizing a video recognition approach, experiments and tests are conducted on the model's inference performance under different input sizes. The test results indicate that the lightweight model maintains recognition performance while achieving real-time inference speeds.

 doi:10.20944/preprints202408.1630.v1

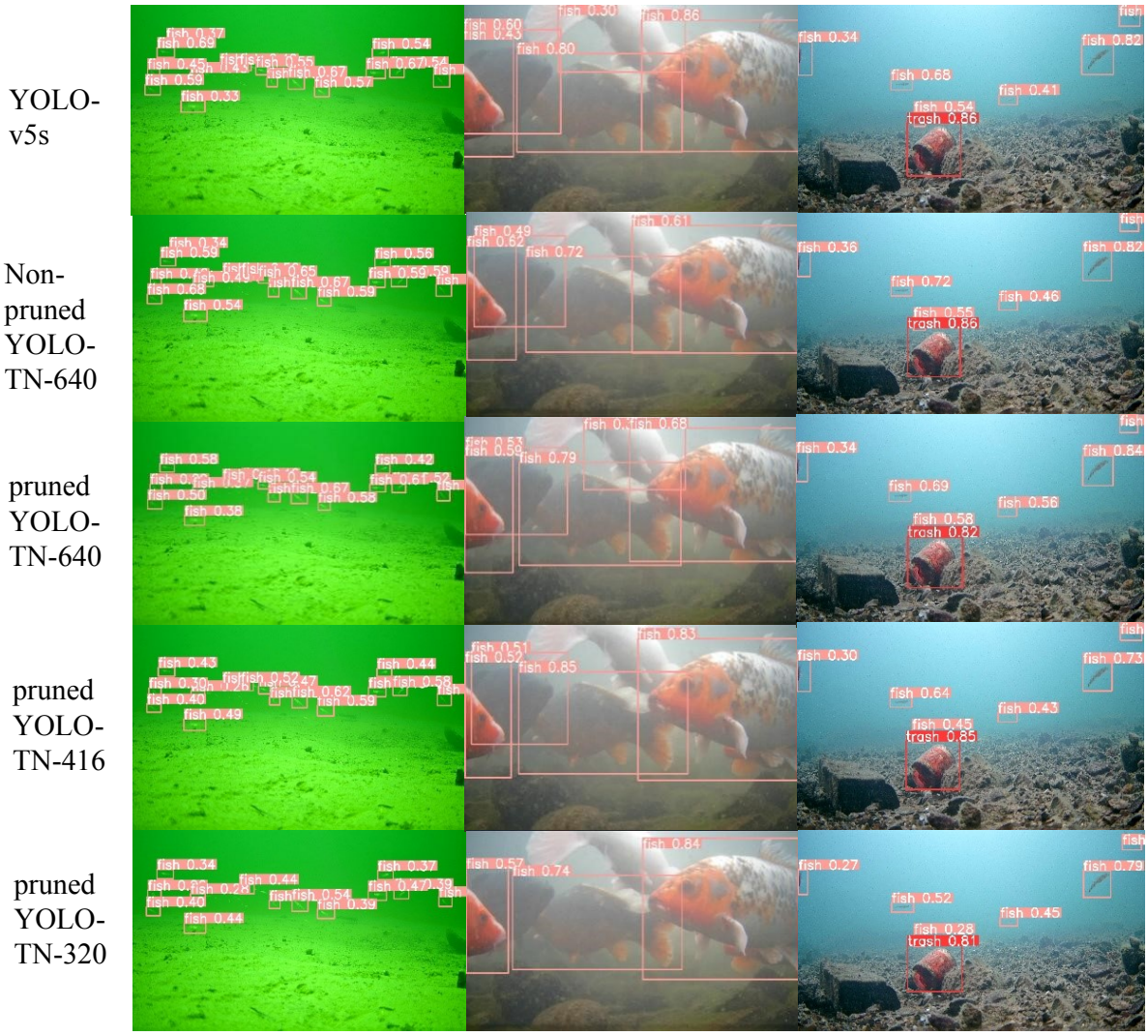


Figure 9. YOLO-TN(e) Complete Structure. The new architecture employs depthwise separable convolution, which divides a standard convolution into depthwise convolution and pointwise convolution, leading to a reduction in the number of parameters and computational workload.

References

1. Wang,Q.;Zeng, X. Deep learning methods and their applications in underwater targets recognition. *Proceedings of the 2015 Academic Conference of the Hydroacoustics Branch of the Acoustical Society of China, Hydroacoustics Branch of the Acoustical Society of China*, **2015**, p. 3. Accessed: Dec. 20, 2023. [Online]. Available:https://kns.cnki.net/kcms2/article/abstract?v=zcLOVLBHd2yuc0K9K0lIzqLOnyKffA5JXrD7S_1b3A_AZXUYyZdd4zqOJi6uoXZuBegPu97bvG__mRmWiZ1qiES5LkrfFdAaLnkYK8_GA9f1_xAZ0NOvmf3X2L4wqsnvfrs4_PiwGj1e4kfoQ9LpLw==&uniplatform=NZKPT&language=CHS
2. Girshick,R.; Donahue,J.; Darrell,T.; Malik,J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. in *2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA: IEEE, Jun*. 2014, pp. 580–587. doi: 10.1109/CVPR.2014.81.
3. R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile: IEEE*, Dec. 2015, pp. 1440–1448. doi: 10.1109/ICCV.2015.169.
4. Redmon,J.; Divvala,S.; Girshick,R.;Farhadi, A.; You Only Look Once: Unified, Real-Time Object Detection. in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA: IEEE, Jun*. 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.

 doi:10.20944/preprints202408.1630.v1

5. Xu,S. et al. PP-YOLOE: An evolved version of YOLO. *arXiv*, Dec. 11, 2022. doi: 10.48550/arXiv.2203.16250.
6. Jocher,G. YOLOv5 by Ultralytics. May 2020. doi: 10.5281/zenodo.3908559.
7. Krizhevsky,A.; Sutskever,I. Hinton,G. E. ImageNet classification with deep convolutional neural networks. *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.
8. Simonyan,K.; Zisserman,A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv*, Apr. 10, 2015. doi: 10.48550/arXiv.1409.1556.
9. Huynh-Thu,Q.; Ghanbari,M. Perceived quality of the variation of the video temporal resolution for low bit rate coding. Accessed: Dec. 20, 2023. [Online]. Available: https://www.researchgate.net/publication/266575823/_Perceived_quality_of_the_variation_of_the_video_temporal_resolution_for_low_bit_rate_coding
10. Han,S.; Pool,J.; Tran,J.; Dally,W,J. Learning both weights and connections for efficient neural networks. in *Proceedings of the 28th International Conference on Neural Information Processing Systems* - Volume 1, in NIPS'15. Cambridge, MA, USA: MIT Press, Dec. 2015, pp. 1135–1143.
11. Wen,W.; Wu,C.; Wang,Y.; Chen,Y.; Li,H. Learning structured sparsity in deep neural networks, in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, in NIPS'16. Red Hook, NY, USA: Curran Associates Inc., Dec. 2016, pp. 2082–2090.
12. Lin,M. et al. HRank: Filter Pruning Using High-Rank Feature Map. in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 1526–1535. doi: 10.1109/CVPR42600.2020.00160.
13. Gao,S.; Huang,F.; Cai,W.; Huang,H. Network Pruning via Performance Maximization. in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA: IEEE, Jun. 2021, pp. 9266–9276. doi: 10.1109/CVPR46437.2021.00915.
14. Gholami,A.; Kim,S.; Dong,Z.; Yao,Z.; Mahoney, M. W.; Keutzer,K. A Survey of Quantization Methods for Efficient Neural Network Inference. *arXiv*, Jun. 21, 2021. doi: 10.48550/arXiv.2103.13630.
15. Faraone,J.; Fraser,N.; Blott,M.; Leong,H.W.; SYQ: Learning Symmetric Quantization for Efficient Deep Neural Networks. in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT: IEEE, Jun. 2018, pp. 4300–4309. doi: 10.1109/CVPR.2018.00452.
16. Courbariaux,M.; Hubara,I.; Soudry,D.; REl-Yaniv,.; Bengio,Y. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. *arXiv*, Mar. 17, 2016. doi: 10.48550/arXiv.1602.02830.
17. Chen,P.; Liu, J.; Zhuang,B.; Tan,M.; Shen,C. AQD: Towards Accurate Quantized Object Detection. in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA: IEEE, Jun. 2021, pp. 104–113. doi: 10.1109/CVPR46437.2021.00017.
18. Zhang,X.; Zhou,X.; Lin, M.; Sun,J. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT: IEEE, Jun. 2018, pp. 6848–6856. doi: 10.1109/CVPR.2018.00716.
19. Wang,X.; Kan, M.; Shan,S.; Chen,X. Fully Learnable Group Convolution for Acceleration of Deep Neural Networks. in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA: IEEE, Jun. 2019, pp. 9041–9050. doi: 10.1109/CVPR.2019.00926.
20. Gou,J. Yu,B.; Maybank,S. J.; Tao,D. Knowledge Distillation: A Survey. *Int J Comput Vis*, vol. 129, no. 6, pp. 1789–1819, Jun. 2021, doi: 10.1007/s11263-021-01453-z.
21. Buciluǎ,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: ACM, Aug. 2006, pp. 535–541. doi: 10.1145/1150402.1150464.
22. Zagoruyko,S.; Komodakis,N. Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. *arXiv* Feb. 12, 2017. doi: 10.48550/arXiv.1612.03928.
23. B. Heo, M. Lee, S. Yun, and J. Y. Choi, "Knowledge Transfer via Distillation of Activation Boundaries Formed by Hidden Neurons," *AAAI*, vol. 33, no. 01, pp. 3779–3787, Jul. 2019, doi: 10.1609/aaai.v33i01.33013779.
24. Peng,B. et al. Correlation Congruence for Knowledge Distillation, in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South): IEEE, Oct. 2019, pp. 5006–5015. doi: 10.1109/ICCV.2019.00511.
25. Cho,J.H.; Hariharan, B. On the Efficacy of Knowledge Distillation. in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South): IEEE, Oct. 2019, pp. 4793–4801. doi: 10.1109/ICCV.2019.00489.

 doi:10.20944/preprints202408.1630.v1

26. Mirzadeh,S.I.; Farajtabar,M.; Li,A.; Levine,N.; Matsukawa,A.; Ghasemzadeh,H. Improved Knowledge Distillation via Teacher Assistant. *AAAI*, vol. 34, no. 04, pp. 5191–5198, Apr. 2020, doi: 10.1609/aaai.v34i04.5963.
27. Liu,Y. et al. Search to Distill: Pearls Are Everywhere but Not the Eyes. in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA: IEEE, Jun. 2020, pp. 7536–7545. doi: 10.1109/CVPR42600.2020.00756.
28. Shen,S.H.; Li,Y L.; Qiang,Y.K.; Xue,R.L.; Jun,W.L. Research on Compression of Teacher Guidance Network Use Global Differential Computing Neural Architecture Search. in *2022 5th International Conference on Artificial Intelligence and Big Data (ICAIBD)*, Chengdu, China: IEEE, May 2022, pp. 526–531. doi: 10.1109/ICAIBD55127.2022.9820338.
29. Liu,H.; Simonyan,K.; Yang,Y. DARTS: Differentiable Architecture Search. *arXiv*, Apr. 23, 2019. doi: 10.48550/arXiv.1806.09055.