water

MDPI

Article

Development of a Lightweight Floating Object Detection Algorithm

Rundong Xian, Lijun Tang * and Shenbo Liu

School of Physics and Electronic Science, Changsha University of Science and Technology, Changsha 410114, China; jacklilifg@gmail.com (R.X.)
* Correspondence: tanglj@csust.edu.cn

Abstract: YOLOv5 is currently one of the mainstream algorithms for object detection. In this paper, we propose the FRL-YOLO model specifically for river floating object detection. The algorithm integrates the Fasternet block into the C3 module, conducting convolutions only on a subset of input channels to reduce computational load. Simultaneously, it effectively captures spatial features, incorporates reparameterization techniques into the feature extraction network, and introduces the RepConv design to enhance model training efficiency. To further optimize network performance, the ACON-C activation function is employed. Finally, by employing a structured non-destructive pruning approach, redundant channels in the model are trimmed, significantly reducing the model's volume. Experimental results indicate that the algorithm achieves an average precision value (mAP) of 79.3%, a 0.4% improvement compared to yolov5s. The detection speed on the NVIDIA GeForce RTX 4070 graphics card reaches 623.5 fps/s, a 22.8% increase over yolov5s. The improved model is compressed to a volume of 2 MB, representing only 14.7% of yolov5s.

Keywords: YOLOv5 algorithm; floating object detection; lightweight model

Citation: Xian, R.; Tang, L.; Liu, S. Development of a Lightweight Floating Object Detection Algorithm. *Water* **2024**, *16*, 1633. https://doi.org/10.3390/w16111633

Academic Editors: Chang Huang and Tzu-Yi Pai

Received: 11 April 2024
Revised: 14 May 2024
Accepted: 29 May 2024
Published: 6 June 2024

1. Introduction

Various floating objects frequently appear in water bodies such as rivers, lakes, and reservoirs. Their substantial accumulation can adversely affect water quality and the safe operation of hydraulic engineering [1]. Traditional river drift detection mainly relies on manual inspection, which is labor-intensive and inefficient [2]. Achieving the target detection of floating objects through monitoring videos for automatic identification and processing of river images significantly enhances detection efficiency and saves human resources. Compared to manual detection, target detection based on artificial intelligence is less influenced by subjective factors and can provide more accurate detection results. With the development and application of machine learning technologies, the efficiency and accuracy of floating object detection have gradually improved [1].

In the early stages of floating object detection, methods such as image segmentation [3] and background subtraction [4] were commonly employed. However, these detection approaches often result in issues related to the completeness and accuracy of target extraction, especially in scenarios with fast-moving floating objects and significant variations in scene lighting. This leads to serious problems of missed detections and false negatives, making it challenging to meet the current requirements for floating object detection accuracy.

In recent years, deep learning has become the trend in floating object detection, and this technology has made significant strides in the field. The development of deep learning object detection algorithms has gone through several important stages. Two-stage object detection algorithms: In this stage, algorithms first generate candidate boxes and then classify and regress each candidate box. The R-CNN [5] series represents this stage, which initially uses traditional methods such as selective search to generate candidate boxes, then employs a CNN for feature extraction, and finally uses classifiers like SVM [6] for

classification. Single-stage object detection algorithms: To further improve the speed of object detection, researchers have proposed single-stage object detection algorithms such as YOLO (You Only Look Once) [7] and SSD (Single-Shot MultiBox Detector) [8]. These algorithms do not require generating candidate boxes; instead, they directly perform dense sampling on the image, transforming the object detection task into a regression problem.

Lin et al. [9] achieved river floating object detection based on unmanned aerial vehicles by combining time difference detection and background subtraction. However, due to variations in water surface characteristics, this approach was susceptible to detection errors. M Tharani et al. [10] introduced a new category for garbage detection, providing a manually collected and annotated dataset of garbage images. They proposed a new attention layer focusing on smaller objects and established a deep learning network based on the Darknet framework. F Lin et al. [11] proposed an enhanced YOLOv5s (FMA-YOLOv5s) algorithm by adding a Feature Map Attention (FMA) layer at the end of the backbone network to enhance the network's feature extraction capability. The FMA layer does not alter the size of the input feature map, allowing it to be easily and flexibly added to any network structure. Kong et al. [12] employed a target detection algorithm based on the YOLOv3 network, enhancing detection accuracy and speed by training on a self-constructed dataset, ultimately deploying it for detection on robots. Z Yi et al. [13] proposed a detection and localization algorithm aimed at addressing the issue of low accuracy in unmanned boats when detecting and locating floating objects. Chen et al. [14] enhanced the detection of small floating objects on water surfaces in the YOLOv5 network architecture by incorporating a small target detection head in the shallow layers. This integration of spatial and semantic feature information maximally preserves the crucial features of small floating objects, thereby improving their detection performance. The loss function CIOU was replaced with the SIOU, where the SIOU considers the orientation of the real and predicted frames, enhancing the effectiveness of detecting small floating objects on the water surface.

In terms of making YOLOv5 lightweight, Gangliu et al. [15] introduced the C3Ghost and GhostConv modules into the backbone network of YOLOv5. They also combined the DWConv module with the C3Ghost module in the neck network of YOLOv5 and tested it on the PascalVOC dataset. The algorithm reduced FLOPs by 54% and the number of model parameters by 52.53%, while maintaining the same mAP. Rio Arifando et al. [16] integrated the GhostConv and C3Ghost modules into the YOLOv5 network to reduce the number of parameters and floating-point operations per second (FLOPs). They also added the SimSPPF module to replace the SPPF module in the YOLOv5 backbone network, aiming to improve computational efficiency and accurate object detection capability. The improved algorithm reduced FLOPs by 48% and decreased the number of model parameters by 59.4%. S Chen et al. [17] integrated the GhostConv module into the YOLOv5s network and added the CBAM attention module in the backbone network. They replaced the upsampling module of the network with a lightweight upsampling operator called CARAFE. The improved network reduced the parameter count from 7.1 M to 3.9 M and increased the mAP by 1.5%.

The current mainstream detection algorithms include YOLOv5 [18,19], YOLOv6 [20], YOLOv7 [21], and so on, among others. YOLOv5, developed by Ultralytics in 2020, has now evolved to version 7.0. Despite improvements in the network structure and training strategies introduced in YOLOv6 and subsequent models, YOLOv5 has been extensively validated in practical applications, demonstrating stable and reliable performance. Therefore, this study opted to enhance YOLOv5 for the detection of floating object targets.

Although YOLOv5 performs well on large public datasets such as MSCOCO [22] and PASCAL VOC, it still has the following shortcomings in practical detection tasks, such as floating object detection on video surveillance platforms: (1) Analyzing the fully convolutional network structure of YOLOv5 reveals a series of standard convolutions, upsampling, residual units, and other basic modules. However, as the network deepens, the increase in network parameters and computational costs poses high hardware requirements for platforms deploying and running this algorithm. Therefore, further improvements in the algorithm's model complexity are needed to adapt to more lightweight hardware

platforms. (2) Floating objects exhibit significant scale variations, including many small targets. YOLOv5's multi-scale prediction approach lacks robustness in feature extraction, especially leading to suboptimal recognition of small-scale floating object targets. This paper focuses on addressing the shortcomings in the YOLOv5 network structure and proposes a lightweight floating object detection algorithm based on YOLOv5.

2. Lightweight Approaches to the YOLOv5 Algorithm

2.1. YOLOv5 Algorithm

The YOLOv5 algorithm incorporates the concept of grids, dividing the image into multiple segments. Each grid is assigned the task of predicting one or more objects, generating prediction boxes. The ability of grids to generate prediction boxes stems from the existence of several template prediction boxes, referred to as "anchors", within each grid. Each anchor is characterized by predefined width, height, coordinates, and confidence values. During the training process, if the center of a manually annotated box falls within a particular grid, the corresponding anchor undergoes vigorous "growth" or "shrinkage" towards the annotated box, setting its confidence to 1 (indicating the presence of an object). In contrast, anchors in grids without predicted boxes are assigned a confidence of 0. The objective detection problem is thus simplified to a combination of regression prediction and classification issues, treating the differences in width, height, and coordinates between anchors and real boxes as loss, and employing binary cross-entropy as the confidence loss function.

YOLOv5 incorporates PANet (Path Aggregation Network) [23] in its network structure to construct a Feature Pyramid [24], as shown in Figure 1. The Feature Pyramid aids in detecting objects at different scales, thereby enhancing the model's robustness to objects of various sizes. The backbone network adopts the CSPDarkNet53 + Focus structure, which effectively extracts features but complicates the structure, leading to an increase in parameters and hindering the process of making the model lightweight. The feature extraction network adopts the SPP + PAN structure, further enhancing the network's feature extraction capabilities, improving the model's accuracy in identifying different objects but increasing complexity and computational load, thereby slowing down the inference process. YOLOv5's activation function utilizes the SilU function, a non-linear activation function with smooth and non-monotonic characteristics. However, it may lack sufficient discriminative power in certain situations, limiting its ability to enhance model performance.

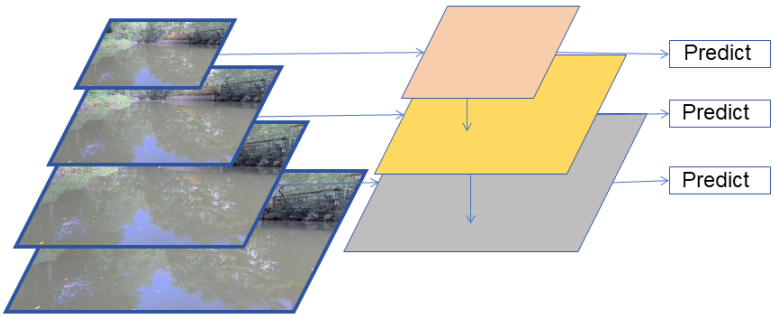


Figure 1. Feature Pyramid structure.

This paper primarily makes improvements to the structure of YOLOv5 in three aspects, constructing the FRl-YOLOv5s model, as shown in Figure 2. Firstly, it extracts the FasterNet block from FasterNet [25] and integrates it into the C3 module of the YOLOv5 feature extraction network, forming C3-Faster. Simultaneously, it applies RepConv, proposed in the RepVGG [26] network, to the YOLOv5 feature fusion network to replace the Conv module, reducing the parameter count while enhancing feature fusion capabilities. Secondly, it employs the ACON (Activate or Not) activation function to replace the SiLU activation function originally used in the C3 module of YOLOv5. ACON can adaptively choose whether to activate neurons, further improving the model's precision and robust-

ness. Finally, it utilizes the LAMP algorithm to perform structured pruning on the model, achieving significant optimization in accuracy and volume for the improved model.

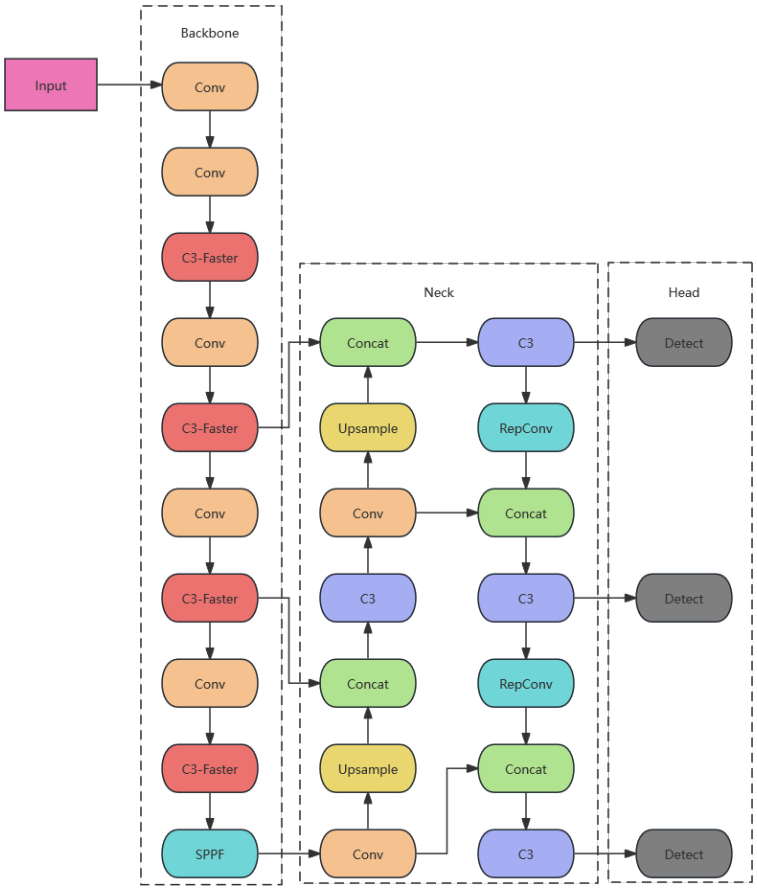


Figure 2. Network architecture of FRL-YOLO.

2.2. C3-Faster

FLOPS (floating-point operations per second): FLOPS represents the number of floating-point arithmetic operations that the system can perform within one second. FLOPs: FLOPs represent the actual number of floating-point arithmetic operations performed by the system or algorithm. Reducing FLOPs can alleviate the computational burden of neural networks, thereby shortening both forward and backward propagation times. This reduction significantly contributes to achieving lower latency. Simultaneously, a lower number of FLOPs can enhance the parallelism of neural networks on hardware, improving throughput and accelerating the processing speed of input data streams. Therefore, to achieve lightweight neural networks that meet the demands for low latency and high throughput, previous research has predominantly measured the reduction in FLOPs (SqueezeNet [27], GhostNet [28], MobileNets [29]). However, there is a relationship between neural network latency and FLOPs as follows (refer to Equation (1)):

$$\text{Latency} = \frac{\text{FLOPs}}{\text{FLOPS}} \tag{1}$$

However, while reducing FLOPs, it leads to increased memory access, accompanied by additional segmented computations (connection, shuffling and pooling, normalization, and activation). Previous focus was solely on FLOPs, with limited attention to FLOPS. Although some awareness existed, it was considered that there were no viable alternatives. Research indicates that the main cause of the low-FLOPS issue is frequent memory access [25]. Therefore, this paper introduces a novel CSP architecture, C3-Faster, to address this problem.

The traditional C3 module is the main module for learning residual features, as shown in Figure 3. Its structure consists of two branches: one utilizes specified multiple Bottleneck stacks and three standard convolutional layers, while the other passes through a basic convolutional module. Finally, the outputs of both branches are concatenated. This paper replaces the Bottleneck in the C3 module of the YOLOv5 backbone network with PConv, and the branches undergo residual operations, forming the C3-Faster module, as shown in Figure 4. PConv uses the standard Conv module only on certain input channels for convolutional operations, enabling spatial feature extraction. This method allows selective processing while retaining specific channel information. The FLOPs calculation for PConv only (refer to Equation (2)) is:

$$h \times w \times k^2 \times c_p^2 \tag{2}$$

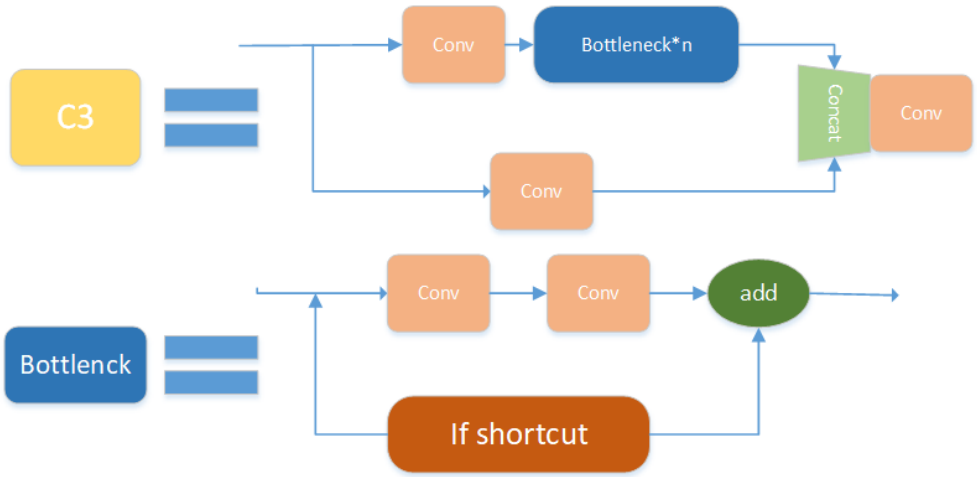


Figure 3. C3 Module.

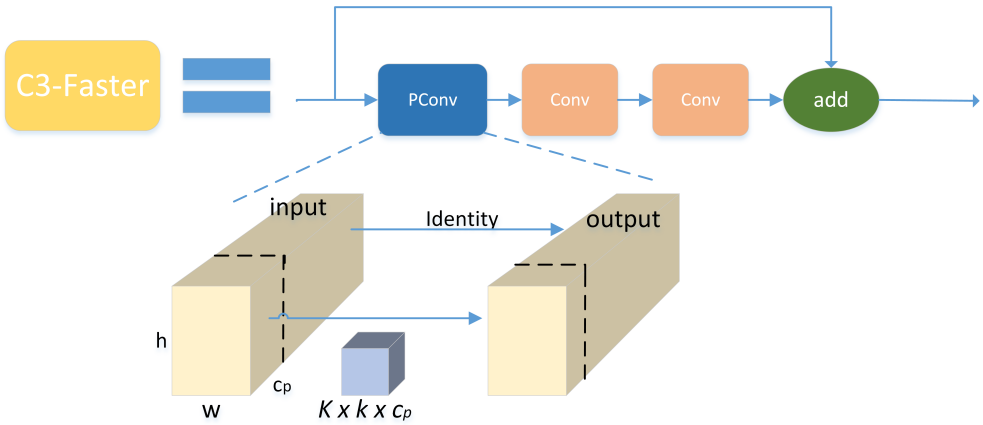
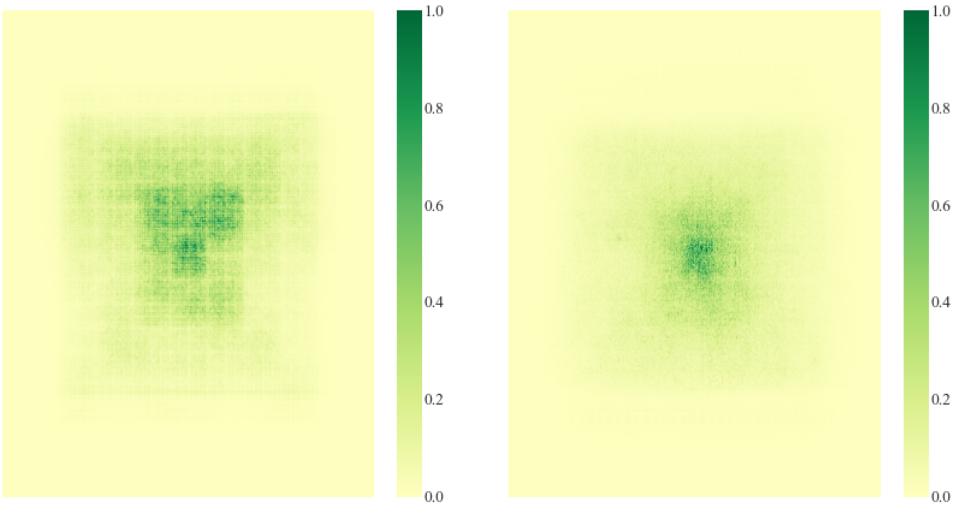


Figure 4. C3-Faster Module.

In this context, c_p denotes the number of channels involved in regular convolutions within PConv, typically considered as 1/4 of c. Consequently, the floating-point operations per second (FLOPs) of PConv are only 1/16 of those in a standard convolution, leading to a reduction in memory access. This substitution enables a decrease in the frequency of neural network memory access. Such an alternative allows the maintenance of a higher number of FLOPS with fewer FLOPs, thereby reducing neural network latency. And we believe that through C3-Faster, FRL-YOLO has a large receptive field which will more effectively extract spatial features, as compared in Figure 5.



(**a**) (**b**)

Figure 5. The backbone Effective Receptive Field (ERF) of FRL-YOLO and YOLOv5s. A more widely distributed dark area indicates a larger ERF. FRL-YOLO clearly has a larger ERF. This demonstrates the robustness of feature extraction. (**a**) FRL-YOLO. (**b**) YOLOv5s.

2.3. RepConv

RepVGG is a technique designed for model reparameterization, enabling the transformation of a complex convolutional neural network into a network composed of simple convolutional and fully connected layers. This reparameterization process significantly reduces the computational workload and the number of parameters in the model, thereby enhancing the inference speed and lightweight deployment capabilities. The core concept involves replacing the original convolutional operations with a module consisting of convolutional layers and element-wise addition operations, achieving the reparameterization of the network.

Integrating RepVGG into the feature fusion network involves constructing the training-time Feature Replication Layer (FRL) using the identity branch, 1×1 convolution, and 3×3 convolution. During the inference phase, structural reparameterization transforms the identity branch, 1×1 convolution, and 3×3 convolution into a 3×3 RepConv block, as shown in Figure 6. The multi-branch topology architecture learns diverse feature information during training, while the simplified single-branch architecture saves memory consumption during inference, enabling rapid inference. Following multi-branch training on one tensor, it is channel-wise concatenated with another tensor. Channel-wise random operator is also employed to enhance information fusion between the two tensors, achieving deep integration of features from different input channels with lower computational complexity.

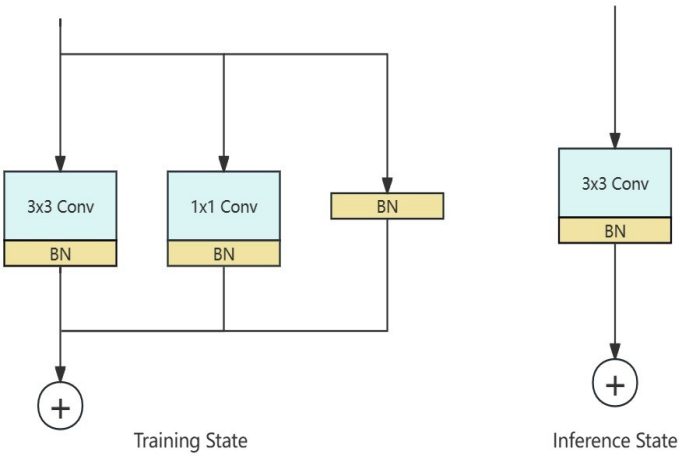


Figure 6. Structural reparameterization of a RepConv block.

2.4. Activation Function Selection

ACON-C is a variant of the ACON (Activate Or Not) activation function. By introducing learnable parameters, the ACON-C activation function effectively models and captures the non-linear relationships within the data, thereby enhancing the expressive power of the model. Additionally, this activation function exhibits the capability to adaptively adjust its shape and characteristics based on the learned parameters. This adaptive feature enables the model to better accommodate diverse input data distributions, ultimately improving the generalization performance of the model. The ACON-C activation function can be expressed using the following form:

$$f_{\text{Acon-c}}(x) = S_\beta(p_1 x, p_2 x) = (p_1 - p_2) * \sigma[\beta(p_1 - p_2)x] + p_2 x \tag{3}$$

In the ACON-C formula, σ represents the sigmoid function, S_β represents the Smooth Maximum, a smoothed and differentiable variant of the MAX function. Here, β represents a smoothing factor. As β approaches infinity, the Smooth Maximum converges to the standard MAX function. Conversely, when β is 0, the Smooth Maximum becomes an arithmetic mean operation; the S_β formula is shown in Equation (4). In the ACON-C formula, the adaptive adjustment of p_1 and p_2 is achieved through two learnable parameters. Taking the first derivative of these parameters yields Equation (5).

$$S_\beta(x_1, \cdots, x_n) = \frac{\sum_{i=1}^{n} x_i e^{\beta x_i}}{\sum_{i=1}^{n} e^{\beta x_i}} \tag{4}$$

$$\lim_{x \to \infty} \frac{df_{\text{Acon}-c}(x)}{dx} = p_1 \quad \lim_{x \to -\infty} \frac{df_{\text{Acon}-c}(x)}{dx} = p_2 \tag{5}$$

As x approaches positive infinity, the gradient of the function is p_1, and as x approaches negative infinity, the gradient is p_2. Taking the second derivative of the function yields Equation (6).

$$\frac{d^2}{\mathrm{dx}^2}[f_{\text{Acon}-c}(x)] = \frac{\beta(p_1 - p_2)^2 \mathrm{e}^{\beta}\left((\beta(p_1 - p_2)x + 2)\mathrm{e}^{\beta(p_1 - p_2)x} + \beta(p_1 - p_2)x + 2\right.}{\left(\mathrm{e}^{\beta(p_1 - p_2)x} + 1\right)^3} \tag{6}$$

In order to obtain the upper and lower bounds for the first derivative, setting the second derivative to zero and solving for the critical points result in the upper and lower bounds for the first derivative, as expressed in Equations (7) and (8), respectively.

$$\text{maxima}\left(\frac{\mathrm{d}}{\mathrm{dx}}[f_{\text{Acon}-c}(x)]\right) \approx 1.0998\mathrm{p}_1 - 0.0998\mathrm{p}_2 \tag{7}$$

$$\text{maxima}\left(\frac{\mathrm{d}}{\mathrm{dx}}[f_{\text{Acon}-c}(x)]\right) \approx 1.0998\mathrm{p}_2 - 0.0998\mathrm{p}_1 (\beta > 0) \tag{8}$$

The upper and lower bounds of the first derivative of ACON-C can be observed to be jointly determined by adjusting two parameters, namely p_1 and p_2. This design imparts a heightened level of flexibility to the model during the training process, enabling it to adapt more dynamically to varying data distributions. By learning and adjusting these two parameters, the model can finely tune the upper and lower bounds of the first derivative, thereby enhancing its capability to capture non-linear relationships within the data. This aspect is crucial for elevating the model's expressiveness and adaptability, underscoring its significance in improving overall performance.

2.5. Model Compression

Currently, popular model compression methods include knowledge distillation [30], parameter quantization [31], and model pruning [32]. This research employs model pruning as the chosen model compression technique. During the pruning process, the focus is primarily

on trimming neurons in deep models and pruning the network model in deep networks. Pruning is generally categorized into structured pruning and unstructured pruning [33]. In this experiment, the structured pruning approach was adopted. In comparison to unstructured pruning, structured pruning typically involves pruning at higher levels, such as layer or filter levels, rather than at the individual weight level. This method constitutes a coarse-grained pruning approach, yet it proves to be a more effective strategy. A notable advantage of this method is its direct compatibility with existing deep learning frameworks and hardware accelerators, eliminating the need for special software or hardware libraries.

This paper utilizes the Layer-Adaptive Magnitude-Based Pruning (LAMP) algorithm for structured pruning [34]. The core of the algorithm involves calculating the LAMP score for each connection and pruning the connection with the minimum score. This algorithm achieves higher sparsity while preserving model performance, thereby contributing to a reduction in computational requirements and memory demands of the model. The LAMP score for the u-th index of the weight tensor W is then defined as:

$$\text{score}(u; W) := \frac{(W[u])^2}{\sum_{v \geq u}(W[v])^2} \tag{9}$$

The LAMP score quantifies the relative importance of the target connection among all surviving connections within the same layer. Once the LAMP scores have been computed, the weights are sorted in ascending order based on a given index map. Subsequently, connections with the minimum LAMP scores are globally pruned until the desired level of overall sparsity constraint is achieved. Notably, the LAMP score does not involve any hyperparameters that require tuning and only necessitates basic tensor operations. This ensures that LAMP scores can be computed with minimal computational overhead.

3. Experimental Setup and Results Analysis

3.1. Experimental Environment

Model training and algorithm configuration were conducted on a desktop computer. The specifications of the experimental environment are shown in Table 1.

Table 1. Experimental environment configuration.

Name	Related Configuration
CPU	Intel Core I5-13490F (Intel Corporation, Santa Clara, CA, USA)
GPU	Nvidia Geforce RTX4070 (Nvidia Corporation, Santa Clara, CA, USA)
GPU Accelerate library	CUDA 11.8
Deep learning framework	Pytorch 2.0
Operating system	Windows 10

3.2. Comprehensive Evaluation Metrics

To comprehensively validate the performance of the lightweight algorithm proposed in this paper, seven metrics were selected for a holistic evaluation of the model, namely Precision, Recall, mean Average Precision (mAP), parameters, model volume, Frames Per Second (FPS), and GFLOPs (Giga Floating-Point Operations Per Second). Precision measures the proportion of true positive predictions among samples predicted as positive by the model, indicating the accuracy of positive predictions. Its expression is given by Equation (10). Recall, on the other hand, assesses the proportion of true positive samples among all actual positive samples, indicating how many true positive samples the model has identified, as expressed in Equation (11).

$$\text{precesion} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{10}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{11}$$

Among them, TP (true positive) indicates the number of correctly predicted positive instances, and FN (false negative) indicates the number of incorrectly classified positive instances; FP (false positive) is the number of negative instances classified as positive categories. Mean Average Precision (mAP) is a weighted average of the Average Precision (AP) across all categories and serves as a comprehensive measure of the model's performance across all classes. In this study, as there is only one detection category, mAP is equivalent to AP. The evaluation metric employed is mAP50, where the "50" denotes an Intersection over Union (IoU) threshold set at 0.5. In the context of object detection tasks, a predicted bounding box is considered correct if its IoU with the ground-truth bounding box is greater than or equal to 0.5. Therefore, mAP50 signifies the average AP across all categories at an IoU threshold of 0.5.

3.3. Experimental Dataset

The experiment utilized the FloW-Img dataset [35], a subset of the FloW dataset and the world's first dataset for an unmanned surface vehicle (USV) perspective on surface-floating garbage detection. This dataset was curated by collecting imagery data at a rate of 10 Hz using an unmanned surface vehicle, followed by downsampling and removal of blurry images to obtain the final dataset. FloW-Img comprises 2000 images with 5271 annotated targets. Notably, more than half of the targets are small objects (area $< 32 \times 32$). The dataset was divided into training and testing sets with a ratio of 3:2. The training settings were a batch size of four, an input image resolution of 640×640, a number of epoch of 200, and a training set size of 1200. The hyperparameter settings used were an initial learning rate of 0.01, a learning momentum of 0.937, a weight-decay of 0.0005, and an IoU training threshold of 0.5.

4. Results

In Tables 2–4, the evaluation data were obtained by training and testing on the FloW-Img floating object dataset. Analyzing the data allowed for the assessment of the performance of both the baseline and optimized models. YOLOv5s* denotes the YOLOv5s model optimized with the C3-Faster module, RepConv module, and the ACON-C activation function.

Table 2. Performance evaluation of different models.

Model	mAP (%)	Parameters (M)	Volume (MB)	GFLOPs	FPS
Faster RCNN	0.481	137.4	108	370.2	281.3
SSD	0.694	26.2	90.6	62.7	256.1
YOLOv3-tiny	0.743	8.6	17.42	12.9	562.6
YOLOv5s	0.789	7.2	14.8	16.4	507.7
YOLOv6s	0.772	18.5	40.6	45.17	448.5
YOLOv7-tiny	0.753	6.0	12.3	13.0	412.3
YOLOv8s	0.807	11.1	22.5	28.4	476.1
YOLOv5s*	0.791	6.4	13.1	13.9	465.9
FRL-YOLO	0.793	0.8	2	4.6	623.5

Table 3. The results of the ablation experiments conducted on YOLOv5, wherein the modules C3-Faster, RepConv, and ACON-C were sequentially inserted to observe their impact on model performance. These experiments aimed to validate the rationality of integrating C3-Faster, RepConv, and the Acon-C activation function into the original YOLOv5 network model.

	C3-Faster	RepConv	ACON-C	P (%)	R (%)	mAP (%)	Volume (MB)	Parameters (M)
A				0.834	0.726	0.789	13.6	7.2
B	√			0.832	0.72	0.775	12.3	6.3
C		√		0.844	0.731	0.785	13.8	7.1
D			√	0.822	0.735	0.794	13.7	7.0
E	√	√		0.829	0.733	0.79	13.1	6.4

Table 3. *Cont.*

	C3-Faster	RepConv	ACON-C	P (%)	R (%)	mAP (%)	Volume (MB)	Parameters (M)
F	√		√	0.824	0.723	0.782	12.4	6.3
G		√	√	0.826	0.732	0.783	13.9	7.1
H	√	√	√	0.839	0.721	0.793	13.1	6.4

Table 4. Results of pruning experiments under different pruning rates.

Model	Pruning Rate (%)	mAP (%)	Parameters (M)	Volume (MB)
YOLOv5s*		0.793	6.4	13.1
YOLOv5s*	40	0.796	2.1	4.6
YOLOv5s*	50	0.796	1.4	3.1
YOLOv5s*	60	0.795	1.0	2.4
YOLOv5s*	70	0.793	0.8	1.9
YOLOv5s*	75	0.782	0.6	1.5
YOLOv5s*	80	0.779	0.5	1.3

As shown in Table 2, the YOLOv5s algorithm was compared with Faster RCNN, SSD, YOLOv3-tiny, YOLOv6s, and YOLOv7-tiny. The mAP values of the YOLOv5s algorithm increased by 30.8%, 9.5%, 5.6%, 2.7%, and 3.6%, respectively, compared to these models. When comparing the YOLOv5s algorithm with YOLOv8s, although YOLOv8s had a 0.18% higher accuracy than YOLOv5s, the parameter count of YOLOv5s was only 64.8% of YOLOv8s, which aligned more with the lightweight principle of this article. Therefore, YOLOv5s was chosen as the benchmark model.

The model size of FRL-YOLO was 2 M, the number of GFLOPs was 4.6 G, and the mAP reached 79.3%. Our model achieved an mAP that surpassed almost all other classic detection models (except for YOLOv8s). Additionally, some YOLO models with similar mAP values to FRL-YOLO, such as YOLOv5s and YOLOv8s, had a computational intensity more than six times that of FRL-YOLO. On the other hand, FRL-YOLO achieved an FPS of 623.5 with a batch size of 128, surpassing all other models in detection speed. This indicates that our approach meets the requirements for real-time detection. In summary, our model strikes an appropriate balance between speed, accuracy, and computational workload.

As shown in Table 3, experiment A represented the original YOLOv5s algorithm, with an mAP of 0.789, a model volume of 13.9 M, and a parameter count of 7,012,822. Experiments B, C, and D introduced the C3-Faster module into the backbone network, the RepConv module into the neck network, and changed the activation function of the C3 module to ACON-C, respectively. These three experiments showed that introducing C3-Faster significantly reduced the parameter count and model volume, albeit with a considerable decrease in accuracy. However, introducing RepConv had little impact on model performance, while introducing the ACON-C activation function increased the average precision by 0.5%, with minimal changes in parameter count and model volume. Experiments E, F, and G combined these three improvement modules in different combinations. Experiment E combined C3-Faster and RepConv, maintaining accuracy while reducing the parameter count by 8.4% and decreasing the model volume by 1.1 M. Experiment F simultaneously incorporated all three improvement modules, resulting in increased detection accuracy and average precision, with only a 0.4% increase in parameter count compared to the original YOLOv5s algorithm in experiment A. Both experiments E and F demonstrated significant reductions in parameter count and model volume compared to the original YOLOv5s algorithm, while also improving average precision. This is because the C3-Faster module was used to reconstruct the receptive field of the backbone network. This enabled it to receive more feature information. The multi-branch structure of the RepConv module also made feature processing more effective. Additionally, the ACON-C activation function can selectively activate neurons in an adaptive manner,

thereby better adapting to different data distributions and patterns and improving the model's adaptability and generalization ability.

In order to further investigate the impact of the pruning rate on network model performance, we designed comparative experiments with different pruning rates, and the results are shown in Table 4. The baseline network of FRL-YOLO was pruned at pruning rates of 40%, 50%, 60%, 70%, 75%, and 80%, and the pruning results were analyzed.

After fine-tuning training, the performance of the improved models at different pruning rates was improved to varying degrees. However, as the pruning rate exceeded 60%, the average precision of the model gradually decreased. When pruning at higher rates, some important feature weights were pruned, leading to a sharp decline in model accuracy.

Regarding model volume and parameter count, they gradually decreased as the pruning rate increased. When the pruning rate reached 80%, the model volume could be reduced to 1.3 M, only 10.3% of the baseline model. However, its accuracy showed a significant decrease compared to the 60% pruning rate scenario. Taking all factors into consideration, we ultimately chose a pruning rate of 70% as the basic pruning parameter for FRL-YOLO. This significantly reduced the model's storage space while maintaining consistency with the baseline model accuracy, achieving a good balance between accuracy and model size.

To visually demonstrate the superior detection performance of the proposed FRL-YOLO in this study, a comparative analysis was conducted on the test set of the floating object dataset. Special emphasis was placed on evaluating the detection models YOLOv5s and YOLOv8s, with a focus on assessing the effectiveness of FRL-YOLO in detecting small floating objects. The results, as depicted in Figures 7–9 (from top to bottom: YOLOv5s, YOLOv8s, FRL-YOLO), revealed that when detecting relatively large floating objects, the performance of YOLOv5s, YOLOv8s, and FRL-YOLO was comparable, with all models successfully detecting them (with confidence levels above 50%). However, when YOLOv5s and YOLOv8s were tasked with detecting small target floating objects, it was observed that the confidence levels were only around 40%, leading to potential misjudgments. In contrast, FRL-YOLOv5s achieved a confidence level of over 50% when detecting small target objects, indicating its robustness in extracting key information from complex scenes for identifying and localizing small targets.

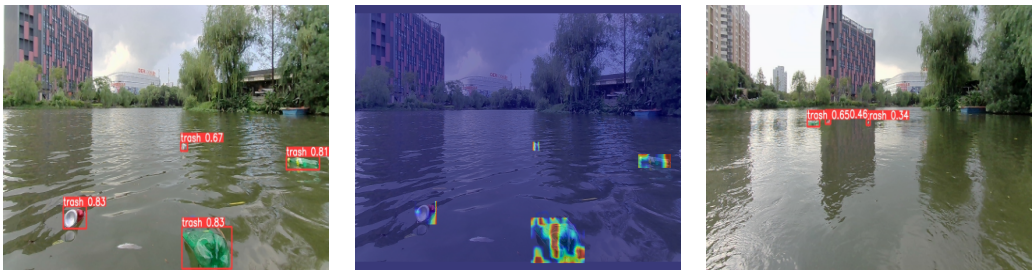


Figure 7. The detection results of YOLOv5s.

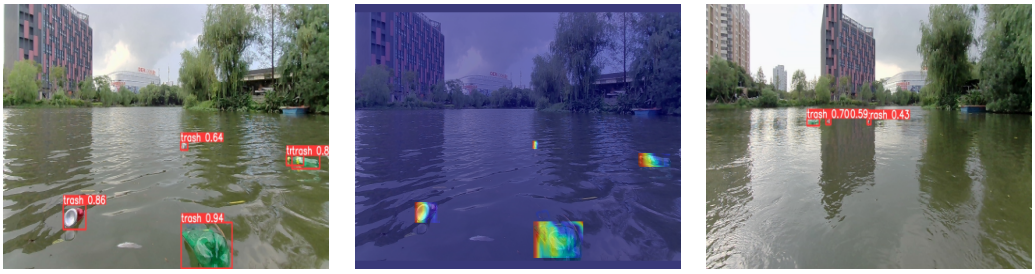


Figure 8. The detection results of YOLOv8s.

In the GradCAM heatmaps shown in Figures 7–9, the color intensity represents the importance of each pixel or region with respect to the predicted target category. Deeper colors (such as red or blue) indicate a greater influence on the prediction results. It is

evident that the YOLOv5s and YOLOv8s models exhibit less attention towards small targets compared to FRL-YOLO, which displays a higher responsiveness towards small targets. This suggests that the FRL-YOLO model effectively focuses on and localizes small targets, even amidst complex scenes.

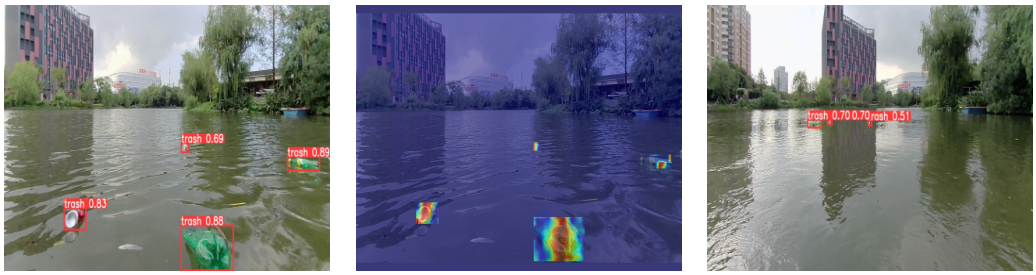


Figure 9. The detection results of FRL-YOLO.

5. Conclusions

The proposed FRL-YOLOv5s algorithm in this paper incorporated several key enhancements. Firstly, it introduced the ACON-C activation function and integrated the C3-Faster module into the feature extraction network of YOLOv5. Additionally, it incorporated the RepConv convolutional operation into the feature fusion network to achieve a lightweight design. Finally, it applied Layer-Adaptive Magnitude-Based Pruning to trim redundant model channels, reducing model volume. Comparative experiments against other YOLO series algorithms such as YOLOv5s, YOLOv7-tiny, and YOLOv8s demonstrated that FRL-YOLOv5s achieved an average detection precision of 79.5%. Furthermore, the model volume was compressed to 1.9M, while achieving an FPS of 623.5. These results underscore the high precision, small model size, and fast inference speed of the FRL-YOLO algorithm, making it a promising candidate for deployment on edge devices.

Author Contributions: Conceptualization, R.X., L.T. and S.L.; methodology, R.X. and S.L.; software, R.X.; validation, R.X.; formal analysis, R.X.; investigation, R.X. and S.L.; data curation, R.X.; writing—original draft preparation, R.X.; writing—review and editing, L.T.; supervision, L.T.; project administration, L.T.; funding, L.T.; acquisition, L.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Van Sebille, E.; Aliani, S.; Law, K.L.; Maximenko, N.; Alsina, J.M.; Bagaev, A.; Bergmann, M.; Chapron, B.; Chubarenko, I.; Cózar, A.; et al. The physical oceanography of the transport of floating marine debris. *Environ. Res. Lett.* **2020**, *15*, 023003. [CrossRef]
2. Van Lieshout, C.; van Oeveren, K.; van Emmerik, T.; Postma, E. Automated river plastic monitoring using deep learning and cameras. *Earth Space Sci.* **2020**, *7*, e2019EA000960. [CrossRef]
3. Shrivakshan, G.; Chandrasekar, C. A comparison of various edge detection techniques used in image processing. *Int. J. Comput. Sci. Issues* **2012**, *9*, 269.
4. McHugh, J.M.; Konrad, J.; Saligrama, V.; Jodoin, P.M. Foreground-adaptive background subtraction. *IEEE Signal Process. Lett.* **2009**, *16*, 390–393. [CrossRef]
5. Bharati, P.; Pramanik, A. Deep learning techniques—R-CNN to mask R-CNN: A survey. In *Computational Intelligence in Pattern Recognition: Proceedings of CIPR 2019*; Springer: Singapore, 2020; pp. 657–668.
6. Supreeth, H.; Patil, C.M. An adaptive SVM technique for object tracking. *Int. J. Pure Appl. Math* **2018**, *118*, 131–135.
7. Jiang, P.; Ergu, D.; Liu, F.; Cai, Y.; Ma, B. A Review of Yolo algorithm developments. *Procedia Comput. Sci.* **2022**, *199*, 1066–1073. [CrossRef]
8. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In *Proceedings of the Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016*; Part I 14; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.

9. Lin, Y.; Zhu, Y.; Shi, F.; Yin, H.; Yu, J.; Huang, P.; Hou, D. Image Processing Techniques for UAV Vision-Based River Floating Contaminant Detection. In Proceedings of the 2019 Chinese Automation Congress (CAC), Hangzhou, China, 22–24 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 89–94.
10. Tharani, M.; Amin, A.W.; Maaz, M.; Taj, M. Attention neural network for trash detection on water channels. *arXiv* **2020**, arXiv:2007.04639.
11. Lin, F.; Hou, T.; Jin, Q.; You, A. Improved YOLO based detection algorithm for floating debris in waterway. *Entropy* **2021**, *23*, 1111. [CrossRef] [PubMed]
12. Kong, S.; Tian, M.; Qiu, C.; Wu, Z.; Yu, J. IWSCR: An intelligent water surface cleaner robot for collecting floating garbage. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *51*, 6358–6368. [CrossRef]
13. Yi, Z.; Yao, D.; Li, G.; Ai, J.; Xie, W. Detection and localization for lake floating objects based on CA-faster R-CNN. *Multimed. Tools Appl.* **2022**, *81*, 17263–17281. [CrossRef]
14. Chen, F.; Zhang, L.; Kang, S.; Chen, L.; Dong, H.; Li, D.; Wu, X. Soft-NMS-enabled YOLOv5 with SIOU for small water surface floater detection in UAV-captured images. *Sustainability* **2023**, *15*, 10751. [CrossRef]
15. Liu, G.; Hu, Y.; Chen, Z.; Guo, J.; Ni, P. Lightweight object detection algorithm for robots with improved YOLOv5. *Eng. Appl. Artif. Intell.* **2023**, *123*, 106217. [CrossRef]
16. Arifando, R.; Eto, S.; Wada, C. Improved YOLOv5-based lightweight object detection algorithm for people with visual impairment to detect buses. *Appl. Sci.* **2023**, *13*, 5802. [CrossRef]
17. Chen, S.; Liao, Y.; Lin, F.; Huang, B. An improved lightweight YOLOv5 algorithm for detecting strawberry diseases. *IEEE Access* **2023**, *11*, 54080–54092. [CrossRef]
18. Jocher, G.; Stoken, A.; Borovec, J.; Changyu, L.; Hogan, A.; Diaconu, L.; Poznanski, J.; Yu, L.; Rai, P.; Ferriday, R.; et al. ultralytics/yolov5: v3.0. Zenodo 2020. Available online: https://zenodo.org/records/3983579 (accessed on 3 April 2024).
19. Li, X.; Li, X.; Han, B.; Wang, S.; Chen, K. Application of EfficientNet and YOLOv5 Model in Submarine Pipeline Inspection and a New Decision-Making System. *Water* **2023**, *15*, 3386. [CrossRef]
20. Li, C.; Li, L.; Jiang, H.; Weng, K.; Geng, Y.; Li, L.; Ke, Z.; Li, Q.; Cheng, M.; Nie, W.; et al. YOLOv6: A single-stage object detection framework for industrial applications. *arXiv* **2022**, arXiv:2209.02976.
21. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2023; pp. 7464–7475.
22. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In *Proceedings of the Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, 6–12 September 2014*; Part V 13; Springer: Berlin/Heidelberg, Germany, 2014; pp. 740–755.
23. Wang, K.; Liew, J.H.; Zou, Y.; Zhou, D.; Feng, J. Panet: Few-shot image semantic segmentation with prototype alignment. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9197–9206.
24. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
25. Chen, J.; Kao, S.h.; He, H.; Zhuo, W.; Wen, S.; Lee, C.H.; Chan, S.H.G. Run, Don't Walk: Chasing Higher FLOPS for Faster Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 12021–12031.
26. Ding, X.; Zhang, X.; Ma, N.; Han, J.; Ding, G.; Sun, J. Repvgg: Making vgg-style convnets great again. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 13733–13742.
27. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. *arXiv* **2016**, arXiv:1602.07360.
28. Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; Xu, C. Ghostnet: More features from cheap operations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2020; pp. 1580–1589.
29. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
30. Wang, L.; Yoon, K.J. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 3048–3068. [CrossRef] [PubMed]
31. Nayak, P.; Zhang, D.; Chai, S. Bit efficient quantization for deep neural networks. In Proceedings of the 2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS), Vancouver, BC, USA, 13 December 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 52–56.
32. Liu, Z.; Sun, M.; Zhou, T.; Huang, G.; Darrell, T. Rethinking the value of network pruning. *arXiv* **2018**, arXiv:1810.05270.
33. Vahidian, S.; Morafah, M.; Lin, B. Personalized federated learning by structured and unstructured pruning under data heterogeneity. In Proceedings of the 2021 IEEE 41st International Conference on Distributed Computing Systems Workshops (ICDCSW), Washington, DC, USA, 7–10 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 27–34.

34. Lee, J.; Park, S.; Mo, S.; Ahn, S.; Shin, J. Layer-adaptive Sparsity for the Magnitude-based Pruning. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 4 May 2021.
35. Cheng, Y.; Zhu, J.; Jiang, M.; Fu, J.; Pang, C.; Wang, P.; Sankaran, K.; Onabola, O.; Liu, Y.; Liu, D.; et al. Flow: A dataset and benchmark for floating waste detection in inland waters. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 10953–10962.