Journal of *Marine Science and Engineering*

MDPI

Article

Sparsity Regularization-Based Real-Time Target Recognition for Side Scan Sonar with Embedded GPU

Zhuoyi Li [1,2], Deshan Chen [3,4,*], Tsz Leung Yip [2] and Jinfen Zhang [3,4,*]

[1] School of Transportation and Logistics Engineering, Wuhan University of Technology, Wuhan 430063, China
[2] Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University, Hong Kong 999077, China
[3] Intelligent Transport Systems Research Center, Wuhan University of Technology, Wuhan 430063, China
[4] National Engineering Research Center Water Transport Safety (WTSC), Wuhan 430063, China
* Correspondence: dschen@whut.edu.cn (D.C.); jinfen.zhang@whut.edu.cn (J.Z.)

Abstract: Side Scan Sonar (SSS) is widely used to search for seabed objects such as ships and wrecked aircraft due to its high-imaging-resolution and large planar scans. SSS requires an automatic real-time target recognition system to enhance search and rescue efficiency. In this paper, a novel target recognition method for SSS images in varied underwater environment, you look only once (YOLO)-slimming, based on convolutional a neural network (CNN) is proposed. The method introduces efficient feature encoders that strengthen the representation of feature maps. Channel-level sparsity regularization in model training is performed to speed up the inference performance. To overcome the scarcity of SSS images, a sonar image simulation method is proposed based on deep style transfer (ST). The performance on the SSS image dataset shows that it can reduce calculations and improves the inference speed with a mean average precision (mAP) of 95.3 and at least 45 frames per second (FPS) on an embedded Graphics Processing Unit (GPU). This proves its feasibility in practical application and has the potential to formulate an image-based real-time underwater target recognition system.

Keywords: side-scan sonar; real-time; target recognition; convolutional neural network (CNN)

Citation: Li, Z.; Chen, D.; Yip, T.L.; Zhang, J. Sparsity Regularization-Based Real-Time Target Recognition for Side Scan Sonar with Embedded GPU. *J. Mar. Sci. Eng.* **2023**, *11*, 487. https://doi.org/10.3390/jmse11030487

Academic Editor: Sergei Chernyi

Received: 26 January 2023
Revised: 17 February 2023
Accepted: 22 February 2023
Published: 24 February 2023

1. Introduction

Side Scan Sonar (SSS) is an equipment that detects the acoustic structure and material properties of a water body and the seabed by radiating sound waves to both sides. Compared with other devices, SSS is more efficient because its operating range is approximately 300–600 m. However, the effective detection range of forward-looking sonar is 2–60 m. Therefore, SSS is mainly used for large-scale underwater scanning search and rescue [1–4]. In the crash of Air France Flight 447 in 2009, rescuers used SSS to identify the wreckage of A380 within an area of 200 × 600 m. After Malaysia Airlines Flight 370 crashed in 2014, rescuers obtained high-resolution seabed images of 710,000 square kilometers with SSS, which is the largest seabed detection result ever [5]. During the long-hour underwater Search and Rescue (SAR) missions, rescuers need to check whether the image contains targets in real-time mode, which is a heavy task and may affect rescue efficiency and even result in omissions.

To shorten rescue time, SSS requires a real-time target detection function. The existing SSS target detection is mainly manual and requires too much manpower in SAR tasks [5,6]. In addition, it is time-consuming to obtain underwater video in advance from the SSS and then load it into the offshore terminal for frame-by-frame inspection. Traditional sonar target recognition algorithms process the acoustic signal directly, which makes them run faster because they do not need to translate the signal into picture, but changes in the acoustic signal invalidate these algorithms [7]. In order to enable the SSS system to have autonomous object detection capability, it is significant to design a real-time target recognition algorithm.

The intractable challenges faced by a real-time SSS object detection system are threefold. First, the environment is complex and variable. Targets to be searched may be at the bottom of a busy river, in a tributary with a lot of silt, or on the deep seabed. Acoustic signals vary greatly in different environments. Second, targets in SSS images may be small because the height of the SSS from the bottom can be up to 300–600 m. Third, computing resources are limited. The SSS equipped on the automatic underwater vehicle (AUV) relies entirely on the batteries for power. To ensure that the SSS works for a long time, the Battery Management System restricts its power consumption [8–10].

Considering that traditional sonar object detection methods rely heavily on the physical characteristics of acoustic signals and do not have good generalization capabilities, deep learning methods, which are more robust in the field of automatic recognition, were introduced [11–13]. Deep learning-based object detection can be divided into one-stage and two-stage methods. Two-stage algorithms first generate location candidate boxes for the input image, and then classify the objects in the candidate boxes, whereas one-stage algorithms simultaneously predict the locations and types of objects. The two-stage methods are in general more accurate but not suitable for autonomous SSS image target recognition systems due to the high computational overhead.

YOLOv5 is a popular one-stage object detection algorithm and is improved from earlier versions, YOLOv4 [14] and YOLOv3 [15]. YOLOv5 takes both accuracy and computational efficiency into account. Test results on a large GPU, the NVIDIA V100, reached a maximum of 55.0 mAP and 400 FPS [16], which is an excellent result, but how to deploy it on a low-power mobile platform has not yet been satisfactorily addressed. There are two main designs for achieving efficient and lightweight target detection. The former strategy reduces the parameters of the network [17,18], the other focuses on introducing low FLOP operations (e.g., group convolution [19] and deep convolution [20]) to reduce the consumption of FLOPs while maintaining the ability to compete with those larger target detection models. However, recent research has shown that fewer parameters and FLOPs do not necessarily lead to faster runs [21]. Furthermore, most of the existing target detection design studies have been evaluated on large GPUs and do not represent how fast models can run on real-time SSS imaging systems.

Further, a large amount of image data is essential to train CNNs. However, the acquisition of SSS images is difficult and expensive. Even if some images are obtained, most of them are meaningless because few images contain true targets such as aircraft wrecks and sunken ships, which makes it difficult for the CNNs to obtain sufficient positive training samples [22].

In this work, a super-efficient target detection model for lightweight GPUs that can be adapted to AUVs is built. First, the fast one-stage algorithm YOLOv5 is used to ensure a high inference speed. However, the large number of parameters of YOLOv5 severely slow down the inference speed on the lightweight GPU. To improve the performance of target recognition while maintaining high efficiency, channel-level sparsity regularization is applied to the target detection model design. By adding penalty terms in the training phase and pruning invalid channels in the inference phase, the channel-level sparsity regularization has proven its effectiveness in several vision tasks. However, the accuracy of a directly pruned network is reduced when performing target detection tasks. To address this issue, an efficient feature encoder, a channel-excitation (CE) feature encoder, is designed. Specifically, CE captures the interactions between channels and efficiently extracts the important small target information by one-dimensional convolution with no more than six parameters. The CE-based target detection model YOLO-slimming not only achieves high target detection accuracy, but also maintains a high inference speed on a lightweight GPU.

In summary, the main contributions of this study are as follows:

(1) A SSS target recognition algorithm is proposed, called YOLO-slimming, based on CNN, which detects underwater objects quickly and accurately with lightweight calculation requirements.

(2) The detection of small single-frame images is investigated in complicated underwater backgrounds. To improve detection accuracy, efficient CE feature encoders are added to the feature extraction module of YOLO-slimming, and multi-scale and weighted image selection training strategies are employed.

(3) To reduce model size, memory overhead and computing consumption, channel-level sparsity regularization is enforced in the training stage and the impact of sparsity on network speed and accuracy is discussed.

(4) A SSS image simulation method based on deep ST is adopted to solve the problem of insufficient data when effective samples are scarce.

2. Relate Work

2.1. SSS Object Detection Based on Deep Learning

Existing SSS target detection designs focus on improving detection accuracy. Ruan et al. [23] introduced residual blocks in the feature extraction phase and designed a Dual-Path Residual network to extract features from noise-correlated targets. Cheng et al. [24] proposed a repeated attention mechanism by combining the channel attention mechanism and the spatial attention mechanism. Song et al. [25] proposed a self-cascaded convolutional module that takes into account both global and local information, which combines both convolutional and cropping layers for feature extraction. The addition of a feature enhancement module to a network improves the performance of the network, but the computational overhead associated with this addition needs to be carefully considered.

Although not as popular as other fields, some scholars have applied YOLO to SSS target detection in recent years. Tang et al. [26] used YOLOv3 to identify sunken ships on the seabed, and achieved a mAP of 89.49% when detecting a single target. Aubard et al. [27] and Li et al. [28] applied YOLOv5 directly to SSS target detection and achieved real-time results. Yu et al. [29] and Sun et al. [30] introduced Transformer, a complex attention mechanism with multiple attention heads, into SSS image target recognition to improve the accuracy of target detection. However, the above experiments were conducted on large GPUs: the effect of real-time operation is achieved by increasing the power consumption of the device, which is not applicable to real-time autonomous SSS target recognition systems. Song et al. [25] and Yu et al. [31] introduced semantic segmentation to accomplish real-time SSS image target recognition. However, as semantic segmentation requires the determination of the classification of each pixel in the image, this method does not offer an advantage in terms of computational speed, considering that the sweep range of a SSS can be up to 300–600 m. In this paper, a target recognition model capable of running in real time on a low-power embedded GPU is presented, which introduces the feature encoder with low parameter counts to improve detection accuracy.

2.2. Computation Reduction

There have also been some attempts to reduce computational costs while maintaining target detection performance.

Low-bit quantization [32,33] represents the parameters the data input to the model in quantized format, which significantly reduces the size and computational cost of the model. However, quantization of object detection models is often poor at maintaining quality due to the high accuracy required for target localization. YOLOv4-Ghost-AMR and YOLOV4-Tinier [34,35] use 1×1 convolutions and depth-separable convolutions to generate redundant images. However, these special operations, while reducing FLOPs, require custom hardware optimizations to achieve high inference speeds, which is not feasible for most commodity mobile devices that only support limited operations. The re-parameterized network uses a multi-branch structure to enable the model to obtain better feature extraction during training, and fuses parallel into serial during testing, thus reducing the computational and parametric quantities and increasing speed [36,37]. However, this approach limits the use of activation functions and reduces the non-linear representation of the model.

Given the limited computational power of the embedded GPUs carried by mobile devices, the fast detection models being developed need to be device friendly.

3. Methods

In this study, a YOLOv5-based model is proposed which can be applied to real-time detection of underwater targets in SSS images. The workflow is shown in Figure 1. First, the optical images containing the targets are collected and synthesized into SSS-style images by ST, which avoids network over-fitting caused by the scarcity of real SSS data. After the expanded data are divided into training sets and test sets, YOLO-slimming is trained, and the trained network is channel-level sparsity regularized to reduce its size. Finally, YOLO-slimming, deployed on an ultralow-power small embedded GPU, is able to detect SSS image targets accurately in real time.

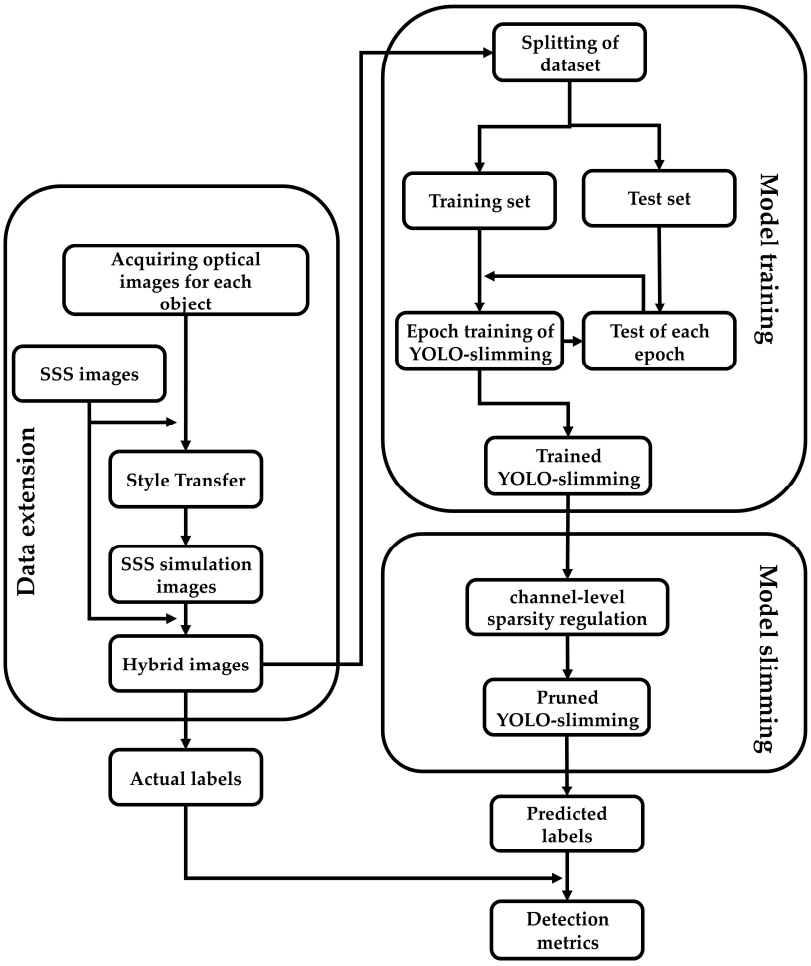


Figure 1. Workflow diagram of YOLO-slimming.

3.1. YOLOv5

YOLOv5 consists of three components: a Backbone module, a Neck module and a Prediction module. The structure of YOLOv5 is shown in Figure 2.

The Backbone module of YOLOv5 is used to extract the features of input images. A Focus module is added at the beginning as a special down-sampling method. The key to Focus is a slicing operation, After Focus, a high-resolution feature map is split into multiple low-resolution feature maps.

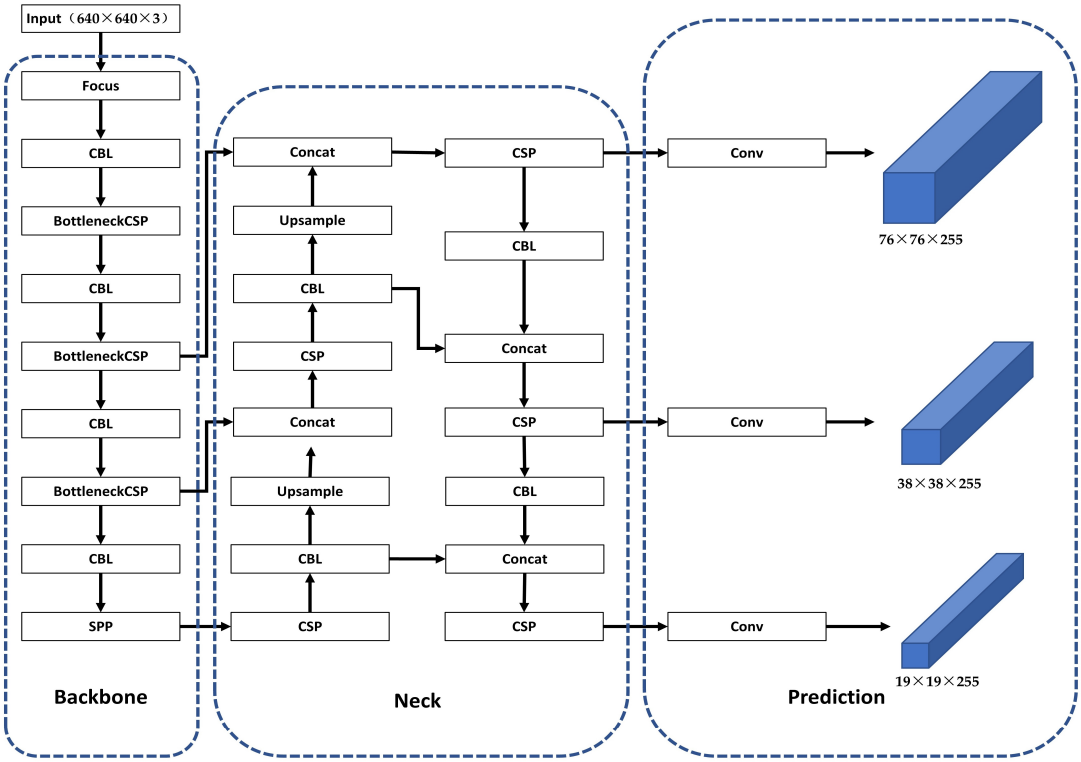


Figure 2. The structure of YOLOv5.

After a Focus module, CBL alternated with a Cross-Stage Partial Network (CSP) [38]. CBL consists of three parts: Convolution, Batch Normalization (BN) and a LeakyRelu activation function. CSP is composed of CBL, Residual and Convolution. CSP integrates the gradient changes into the feature map from beginning to end, which reduces the network parameters and computation, to not only ensure the speed and accuracy, but also reduce the size of the model. CBL and CSP are also widely used in the Neck module to enhance the feature fusion ability.

The last part of the Backbone is the Spatial Pyramid Pool (SPP) [39], which greatly increases the receptive field by concatenating three max pooling layers with different kernel sizes and one non-treated operation in channel dimension. Figure 3 shows the structure of the SPP.

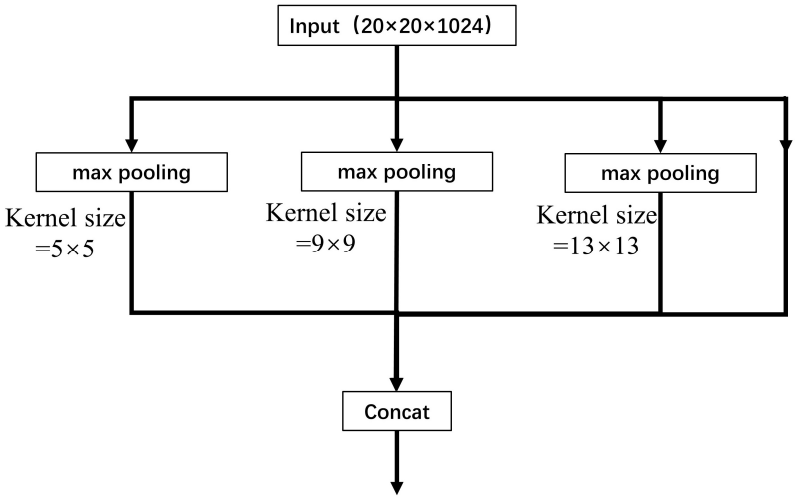


Figure 3. The structure of the SPP.

The Neck module adopts a Path Aggregation Network (PAN) to strengthen feature fusion capability [40]. Figure 4 illustrates a PAN. Up-sampling makes the feature map smaller in size and contains more semantic information. Down-sampling makes the feature map larger and retains more location information. Four different outputs bring robustness to the detection of targets of different sizes.

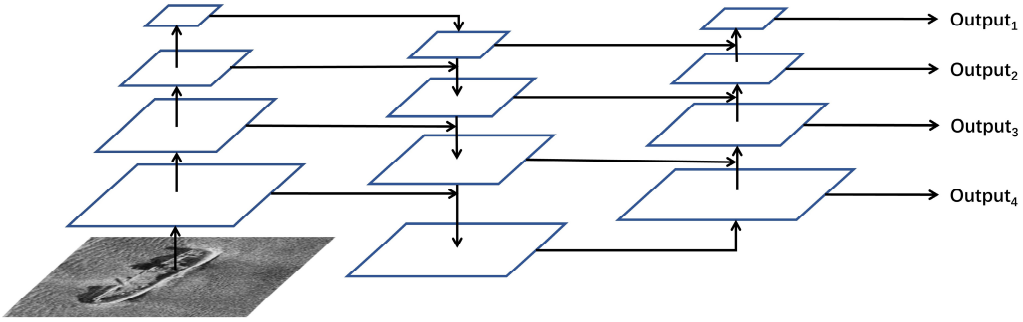


Figure 4. Illustration of a PAN (parallelograms are feature maps).

The Prediction module contains three feature maps with different scales. According to the features at different scales, the corresponding prediction frame is generated for the target image and processed by Non-Maximum Suppression (NMS), and only the prediction frame with the highest local category confidence score is retained [41].

3.2. The CE Feature Encoder

The CE feature encoder is a parallel branch that does not change the structure of Backbone. Specifically, the CE feature encoder includes a global pooling layer, a one-dimensional convolution layer whose kernel size can be adaptively changed, and a sigmoid normalize function [42]. The structure of the CE feature encoder is shown in Figure 5. One-dimensional convolution completes the information interaction cross channels, and the kernel size is determined by Equation (1).

$$k = \frac{\log_2(C) + 1}{2} \tag{1}$$

where C is the number of channels after global pooling; k is the size of the convolution kernel, which represents the coverage of cross-channel interactions.

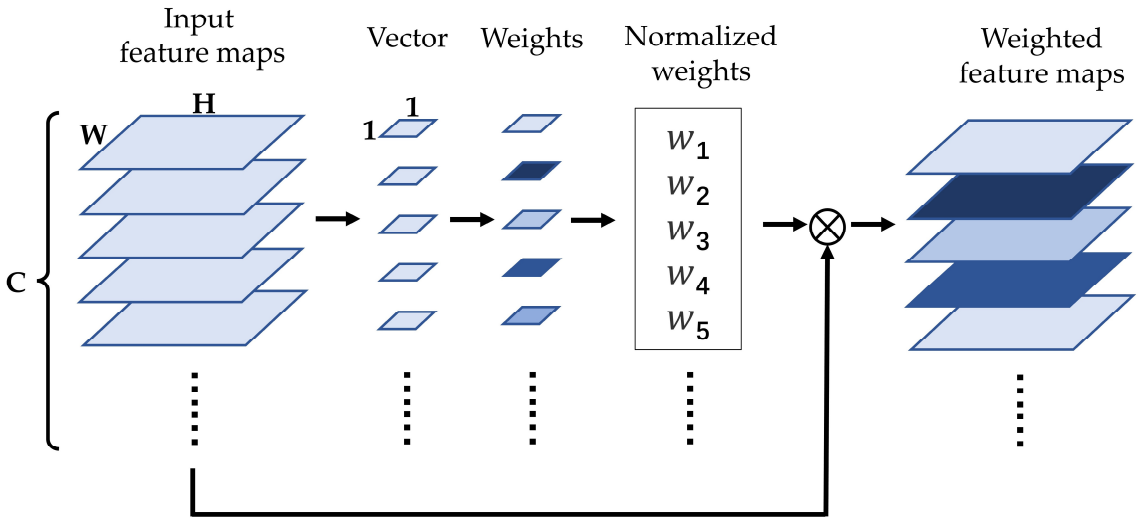


Figure 5. Channel-excitation feature encoder. ⊗ indicates element-wise multiplication.

The CE workflow is divided into the following four steps:

(1) Apply a global average pooling layer to the input feature maps and change them from a matrix of [h, w, c] to a vector with the size of $1 \times 1 \times c$;

(2) Calculate the adaptive one-dimensional convolution kernel size of k;

(3) Apply one-dimensional convolution with kernel size = k to the feature vector to obtain the weights of each channel;

(4) Multiply the normalized weights and the original input feature map channel by channel to generate the weighted feature maps.

Three CE feature encoders are trained and updated together with the last three CBLs of the Backbone module and obtain the importance of each feature map, after which a weight value is signed to each feature map according to its importance. Thus, the neural network can focus on improving the feature maps that are useful for the current task, while suppressing the ones that are not important.

3.3. Channel-Level Sparsity Regularization

To improve the detection accuracy, original YOLOv5 performs multiple convolutions of features in the Backbone, which eliminates some detailed information, especially small targets. To compensate this, YOLOv5 uses the PAN in its Neck module. Three feature maps of different sizes (19×19, 38×38, and 76×76) are fused to retain the necessary information, but it brings much computation in the meantime. To reduce computation, a sparsity regularization called network slimming method is enforced on YOLOv5.

Network slimming generally falls into three categories from a network structure perspective: weight level, layer level and channel level. Weight-level slimming achieves a large compression rate, but specific hardware and libraries are required. The layer-level slimming needs to cut one or more complete layers and is less flexible. In fact, only when the network is deep enough (more than 50 layers), removing layers can bring benefits. Channel-level slimming is a compromise scheme, which is flexible and suitable for any CNNs [40,43].

Scale factor γ is introduced for each channel, which can be multiplied by output features of its corresponding channel. After training, the value of each scaling factor is different. Finally, the scaling factors are pruned below a certain threshold and their corresponding channels. This process is known as channel-level sparsity regularization. Equation (2) is the loss function with the scale factor.

$$L = \sum_{(x,y)} l(f(x,w),y) + \lambda \sum_{channel} |\gamma| \tag{2}$$

where x, y represent the input and target of layer L; W represents the trainable weights in the CNN, the first item corresponds to the original loss function of YOLOv5; $|\gamma|$ is the scale factor corresponding to their channels, which are also trainable parameters. λ is a hyperparameter to balance the two items.

In order not to add additional layer structure, the affine transformation parameters γ of the BN layer is used as the scale factor, which are as follows:

$$\hat{z} = \frac{z_{in} - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{3}$$

$$z_{out} = \gamma \hat{z} + \beta \tag{4}$$

where z_{in} is a mini-batch input into the network, μ and σ are its mean and variance; ϵ is a fixed value which is small enough to prevent the denominator of Equation (3) from being 0. γ and β are the parameters of the affine transformation. γ makes the output value of the network enlarge or reduce, and β makes the output value shift as a whole. γ has the effect of scaling the output of the network. As a parameter of the BN layer, it is trainable itself.

Because of scale factors, the accuracy of the network after sparsity regularization training will decline. Therefore, normal training should be carried out before sparsity

regularization to make the network reach a high accuracy, and a model fine-tuning must be conducted after each channel-level pruning.

3.4. SSS Image Simulation Based on Deep Style Transfer

The application of SSS is mostly related to national defense and underwater rescue, which means there are few SSS images published on the Internet. Furthermore, most SSS images do not include meaningful targets, such as wrecks. Therefore, it is particularly important to simulate more SSS images containing targets. Denos et al. [44] simulated the light shining on an object to obtain an optical image with lateral shadows, then generating a sonar image by adding noise. Based on physical characteristics, it produces high-quality sonar images, but both the simulated image and the target object need to be modeled in a computer. The training of neural networks requires hundreds of images with targets. The workload is therefore too large and the diversity of the modelled objects is not guaranteed.

In order to simulate a large number of SSS images and to ensure the feature diversity of objects in the images at the same time, ST is used to simulate SSS images including targets such as aircraft wrecks and sunken ships. These simulated images are then added to the real SSS image dataset.

Image ST refers to the technology of learning the style of an image (i.e., style image) using a CNN, and then applying to another image (i.e., content image), so that the generated image shares similar styles [45]. ST needs to build a style feature extraction network and a content feature extraction network. Gradient descent is applied after each loop to adjust the generated image to minimize differences in style and content. Equations (5)–(9) are loss functions.

$$L_{\text{content}}\left(\vec{c}, \vec{x}, l\right) = \frac{1}{2}\sum_{i,j}\left(F_{i,j}^{l} - C_{i,j}^{l}\right)^{2} \tag{5}$$

$$G_{i,j}^{l} = \sum_{k=1}^{M_l} N_{ik}^{l} N_{jk}^{l} \tag{6}$$

$$S_{i,j}^{l} = \sum_{k=1}^{M_l} N_{ik}^{l} N_{jk}^{l} \tag{7}$$

$$L_{\text{style}} = \sum_{l} \omega_l \left(\frac{1}{4N_l^2 M_l^2}\sum_{i=1}^{N_l}\sum_{j=1}^{N_l}\left(G_{ij}^{l} - S_{ij}^{l}\right)^{2}\right) \tag{8}$$

$$L_{\text{total}}\left(\vec{c}, \vec{s}, \vec{x},\right) = \alpha L_{\text{content}}\left(\vec{c}, \vec{x}\right) + \beta L_{\text{style}}\left(\vec{s}, \vec{x}\right) \tag{9}$$

where $L_{\text{content}}\left(\vec{c}, \vec{x}, l\right)$ denotes the content loss at layer l; $\vec{c}$, $\vec{x}$ denote the content image and the generated image, respectively; $F_{i,j}^{l}$, $C_{i,j}^{l}$ denote a feature of the generated image and a feature of the content image, respectively. At the ith and jth channel of layer l, $G_{i,j}^{l}$ and $S_{i,j}^{l}$ are Gram matrices of generated image and style image between the ith and jth channels in layer l, respectively; $N_{ik}^{l}N_{jk}^{l}$ denote the kth element of the ith and jth channels, respectively, in layer l; ω_l is the weight factor of each layer to the total loss; M_l denotes the product of the length and width of the feature map.

VGG16 is selected pretrained on the COCO dataset as the network for the above two loss calculations [46]. Compared with other networks, VGG16 processes image features layer by layer, while ResNet [47] and DenseNet [48] implicitly fuse feature maps of different layers. The fusion of introducing shallow information into deeper features increases the computational complexity. The effect of features on ST is analyzed explicitly layer by layer, and the layers involved in loss functions are selected manually to obtain the best visual effect.

We pre-train VGG16 on the COCO dataset because the object categories in COCO contain SSS targets (aircraft, ships and human). Therefore, the pre-trained VGG16 network

can extract the content features of aircraft and ships, while the style features are extracted from the real SSS images.

One loop of SSS image ST with VGG16 is shown in Figure 6. The five CR blocks of VGG16 can be treated as the output of feature extraction, which means that one or more CR blocks can be manually selected to achieve the best ST effect. Figure 7a–e are ST images obtained by calculating losses using CR1–CR5, respectively. The first row is the result of changing content loss, and style loss is updated in the second row.

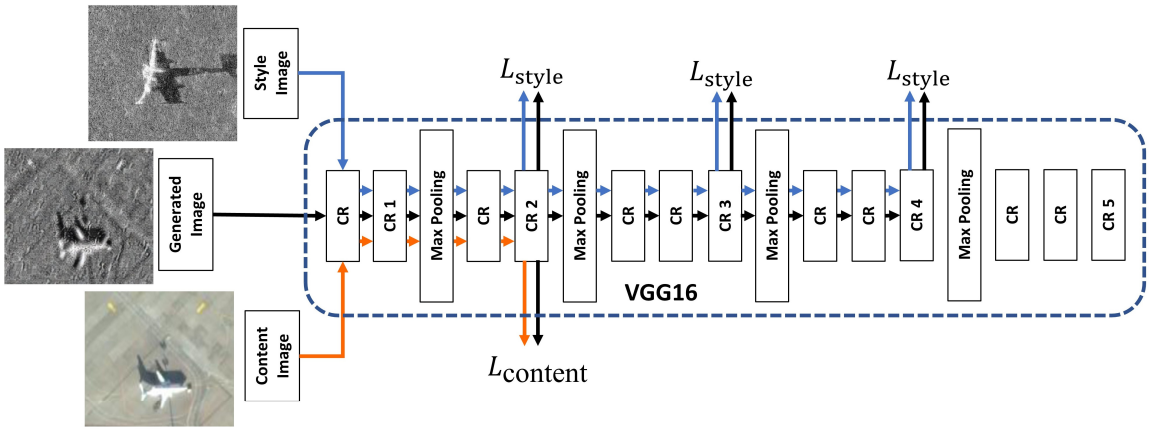


Figure 6. One loop of image ST with VGG16 (generated image is random noise before the first loop).

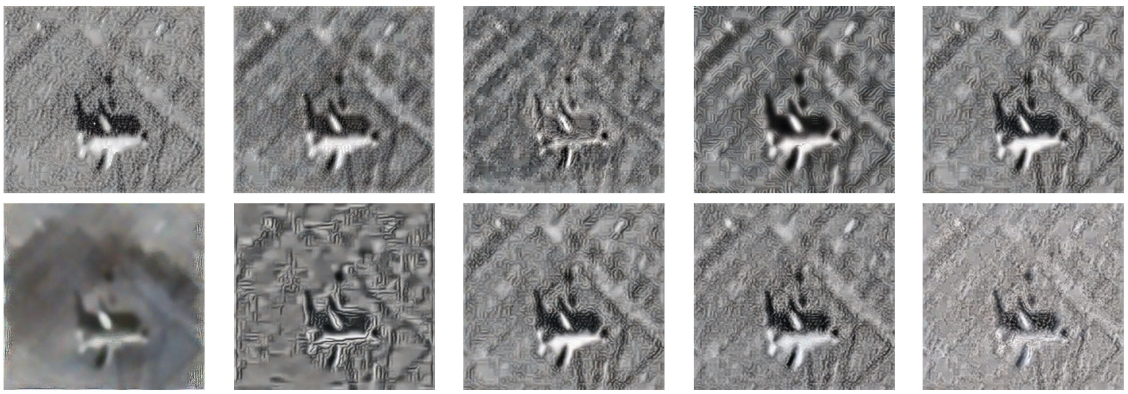


Figure 7. Generated images using different CR blocks as loss. (**a**–**e**) are the ST images obtained by calculating the losses using CR1–CR5, respectively. The first row is the result of changing content loss, and style loss is changed in the second row.

The content loss calculated by CR2 has the highest similarity with the SSS image. The images obtained by using different CR blocks to calculate the style loss are significantly different. With the deepening of the number of CR blocks, the style of the image is closer to the SSS image. The image calculated by CR1 still has an apparent optical image style, and the background of the image calculated by CR2 has inconsistency. The outputs of CR2, CR3 and CR4 are selected to calculate style loss together to obtain the style that is the closest to the SSS image and at the same time have feature diversity.

Figure 8 shows some real SSS images, original optical images, and simulated SSS images after ST. The first column is the SSS images as the style images, the second column is the optical images as the content images, and the third column is the simulated SSS images.

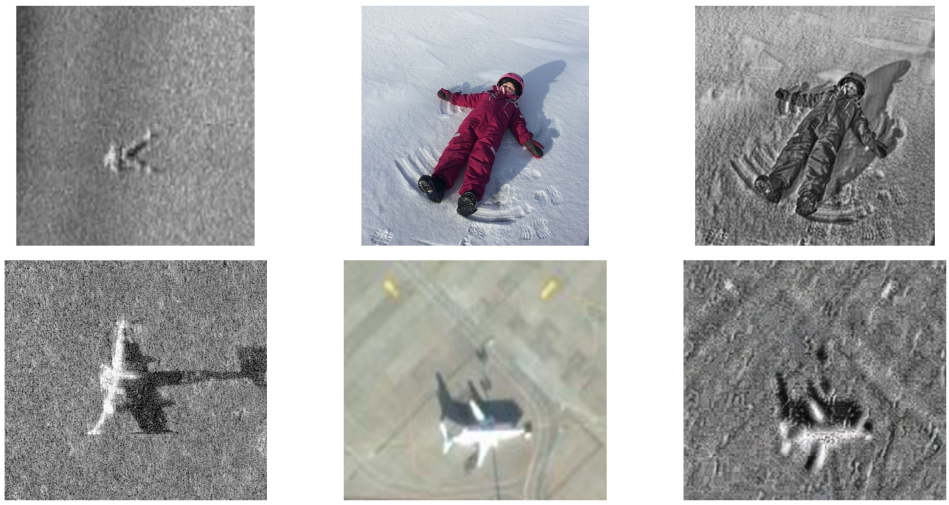


Figure 8. Real SSS images, original optical images and simulated SSS images after ST. The first, second and third columns are style images, optical images and simulated SSS images, respectively.

4. The Dataset and Training Strategy

4.1. The Dataset

The SSS images used in this paper are from Sonar Common Target Detection (SCTD) established by Zhou et al. [49]. The SSS images are divided into three categories: shipwrecks, drowning victims and aircrafts, and the number of images in each category are 385, 90 and 45, respectively. Some images in the dataset are shown in Figure 9. Compared with datasets in the optical field, SCTD has the following characteristics:

- SCTD has collected three types of target images, but the number of samples is unbalanced, especially since the number of images of the drowning victims is only 45.
- The detection range and imaging method of SSS are different, and interference detection such as crushing, fracture and deformation is not satisfactory.
- SCTD is collected from a variety of sources, and image size, resolution, and aspect ratio vary greatly.

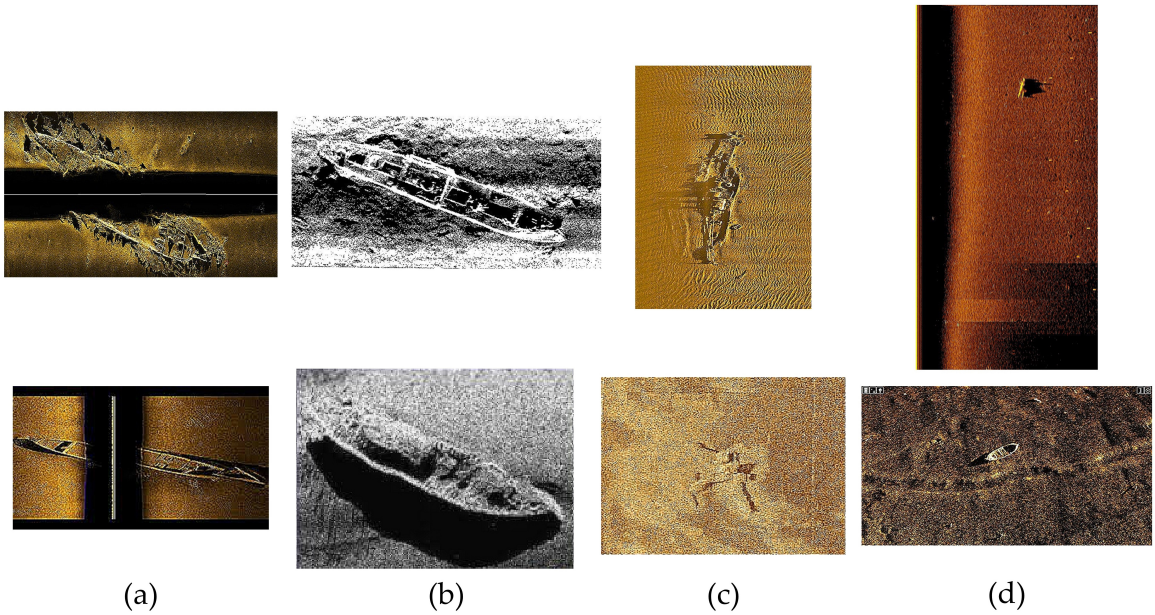


Figure 9. Examples in SCTD: (**a**) fractured; (**b**) low quality; (**c**) eroded; (**d**) small.

Table 1 gives the details of HSCTD. Among the shuffled images, 20% is selected to form a test set to perform cross-validation. Considering that the division of data sets will affect the training effect, it randomly adopts dividing data sets and carries out five repeated experiments. The average value of five experiments is taken as the result of training for analysis.

Table 1. Number of various SSS images.

	Wreck	Airplane	Human
Real images	385	90	45
Simulated images	0	50	90
Hybrid images	385	140	135
Training set	308	112	108
Test set	77	28	27

4.2. Multi-Scale Images

The Backbone of YOLOv5 reduces the size of the input image by a factor of 32, which makes it difficult for the feature descriptions of small objects to be captured by the detection module. At the same time, it is difficult to capture the integrity of oversized targets even after being reduced dozens of times. The same sample is enlarged by a maximum of 1.5 fold and reduced by a minimum of 0.5 fold before being fed into the network, which can improve the robustness of the target detection network to the object size to a certain degree [50]. During training, the original image size is set to 640 × 640. After each epoch, it randomly selects an image size within the set scale range (320 × 320–640 × 640).

4.3. Weighted Image Selection

Although we have supplemented the data for aircraft and drowning victims with ST, the number of shipwrecks is still approximately 3 fold that of the first two. The model will learn more prior information from a high proportion of samples when they are unbalanced, so that the actual prediction will focus on majority, which may lead to better accuracy for many categories, but worse for a small number of categories [51].

The weighted image selection method is as follows:

$$w_i = 1 - \frac{n_i}{\sum n_i} \tag{10}$$

$$P(x,\ w_i) = w_i e^{-w_i x} \tag{11}$$

where n_i is the number of samples of type i, $\sum n_i$ is the sum of the number of labels of all samples, and P is the probability of samples being sampled when the weight is w_i. Equation (11) is a probability density function, and the probability of each sample x being sampled follows exponential distribution.

4.4. Mosaic Image Enhancement

As shown in Figure 9, each HSCTD image contains only 1–2 objects, which cannot meet the requirements of sample diversity. To overcome this problem, mosaic image enhancement is adopted. Mosaic image enhancement randomly cuts and splices four randomly selected images into one image [14]. Figure 10 shows some examples of mosaic image enhancements. By merging four images into one, it can transfer four images to the model at one time, which is equivalent to increasing the batch size. In this way, the background of the detected object is enriched and the robustness enhanced.

Figure 10. Examples of mosaic image enhancement.

4.5. Warm-Up with Cosine Annealing

The learning rate is among the most critical hyper parameters. The large learning rate accelerates the training speed, but with a slow convergence speed [52]. Warm-up training is adopted followed by cosine annealing. At the beginning of training, the model is not stable because it has not been trained enough. The learning rate should be set very low initially, otherwise the model may diverge to disorder. However, a low learning rate makes the training process very slow, so the warm-up training is realized by linearly increasing the low learning rate to a larger value (the set value). The learning rate first increases from zero to the set value, and then attenuates according to Equation (12), the cosine annealing function.

$$\eta_{cur} = \frac{\eta}{2}\left(1 + \cos\left(\frac{E_{cur}}{E}\pi\right)\right) \tag{12}$$

where η_{cur} is the current learning rate; η is the set learning rate; E_{cur} is the current epoch; and E is the epochs required for network training. Since the network does not receive all data at one time and experience one epoch, the value of E_{cur} can be decimal.

4.6. Transfer Learning

Although there are 528 images in the training set together with multi-scale images and mosaic image enhancement, HSCTD is still a small dataset in the field of deep learning. Insufficient samples stop the network from learning enough features, and lead to over-fit. In order to overcome this, transfer learning is adopted [53,54].

When two tasks have similar characteristics, the knowledge between them is helpful to each other's network performance. The COCO dataset has more than 300,000 tag images and 80 categories of targets, including aircraft, ships and people. Therefore, the model is

firstly trained based on the COCO to obtain the pretrained weight, and then HSCTD is used to continue the training process.

4.7. Other Training Details

SGD is used for gradient descent and updating. All models are trained with a set learning rate of 0.01. The batch size is set to 16, and normal training epochs, sparsity regularization training epochs and fine-tuning epochs are 400, 200, and 400, respectively.

5. Experiments and Analysis

5.1. Experimental Conditions

All experiments use desktop computers, in which the computer system is Windows 10, and the experimental environment is shown in Table 2. The total memory of the graphics card is approximately 8192.0 MB. The experimental software uses PyCharm 2021.3, in which the internal environment is PyTorch1.10.1, TensorRT7.2.2 and Python 3.8. Networks are constructed using TensorRT7.2.2 on Jetson Nano.

Table 2. Experimental Environments.

Configuration	Parameters
Operating system for training	Windows 10
CPU	AMD Ryzen7 5800 3.2 GHz
Operating system for inference	Ubuntu 20.04
GPU for training	NVIDIA GeForce RTX2070 8 G
GPU for inference	NVIDIA Jetson Nano 4 G
Accelerated environment	CUDA10.2 + CUDnn7.6.5
Development environment	PyCharm2021
Library	PyTorch1.10.1; TensorRT7.2.2

The evaluation criteria are Precision (P), Recall (R), mAP [55] and FPS. The equation of P and R is as follows:

$$P = \frac{TP}{TP + FP} \tag{13}$$

$$R = \frac{TP}{TP + FN} \tag{14}$$

where TP, TN, FP, and FN represent True Positive, True Negative, False Positive, and False Negative values, respectively.

The mAP is the abbreviation of the mean average accuracy, which is also an indicator to measure the detection accuracy in object detection:

$$AP = \int_0^1 p(r)dr \tag{15}$$

$$mAP = \frac{\sum_{n=1}^{N} AP_n}{N} \tag{16}$$

where p represents Precision, r represents Recall, and p is regarded as a function with r as a parameter; n is the number of object categories, in this study n = 3; AP_n represents the average accuracy of a neural network in identifying a certain type of target. There are two evaluation indexes with mAP, namely mAP@0.5 and mAP@0.5:0.95. They are calculated with lower detection requirements and higher detection requirements, respectively.

The above four indicators are used to measure the accuracy of the neural network. Recognition speed is also a key point. Model size, Params, Floating point of operations (FLOPs) and FPS are used to measure it.

5.2. Network Accuracy Analysis

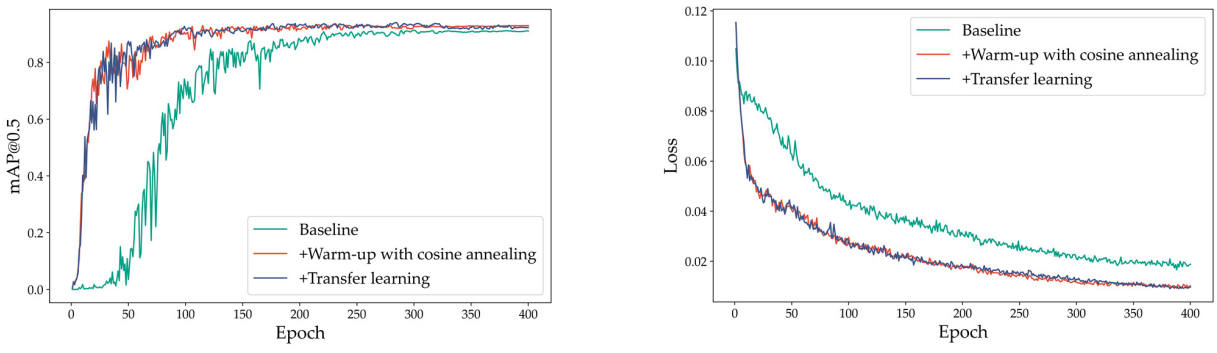
ST is used to expand the dataset, forming HSCTD, and add a CE feature encoder to Backbone to improve the representation of effective feature maps and suppress invalid ones. In the training strategy, we use multi-scale images, weighted image selection, mosaics image enhancement, warm-up with cosine annealing, and transfer learning. The above methods and training strategies are designed to enable the network to fully learn the features of SSS images and to improve the accuracy of recognition. In order to analyze their effects, ablation experiments are conducted. The results are shown in Table 3. The baseline is original YOLOv5, with input image size 640 × 640.

Table 3. Ablation experiments.

Method	*P*	*R*	mAP@0.5	mAP@0.5:0.95
Baseline	88.2%	89.4%	84.9%	71.3%
+HSCTD	93.2%	92.6%	89.6%	73.5%
+CE feature encoder	95.1%	93.9%	89.4%	76.7%
+Multi-scale images	91.6%	92.3%	90.4%	77.6%
+Weighted image selection	92.1%	88.1%	85.9%	74.7%
+Mosaic image enhancement	92.9%	86.3%	88%	69.5%
+Warm-up with cosine annealing	90.5%	89.7%	88.7%	72.9%
+Transfer learning	90.4%	90.4%	87.8%	74.7%
YOLO-slimming (80% pruned)	98.1%	97.2%	95.3%	91.4%

The baseline reached the lowest *P* and mAP@0.5 of 88.2% and 84.9%, respectively. The Recall of the two algorithms adding weighted image selection and mosaic image enhancement is lower than that of the baseline, wherein *R* of weighted image selection is reduced by 1.3%, and *R* of mosaic image enhancement is reduced by 3.1%. In addition, the mAP@0.5:0.95 of mosaic image enhancement is also 1.8% smaller than that of the baseline, reaching 69.5%. The decline in performance may be due to excessive repeated training of some samples caused by weighted image selection. Mosaic image enhancement may also generate images with high truncation ratio, as shown in Figure 10. However, the other behavior of these two algorithms is higher than the baseline, so they still have a positive impact on the model accuracy.

The two training strategies, warm-up with cosine annealing and transfer learning, have limited improvement in P, R and mAP@0.5:0.95, all of which are approximately 1% to 2%. The improvement in mAP@0.5 is less obvious, with approximately 3%. However, they are very helpful to the convergence, as shown in Figure 11.



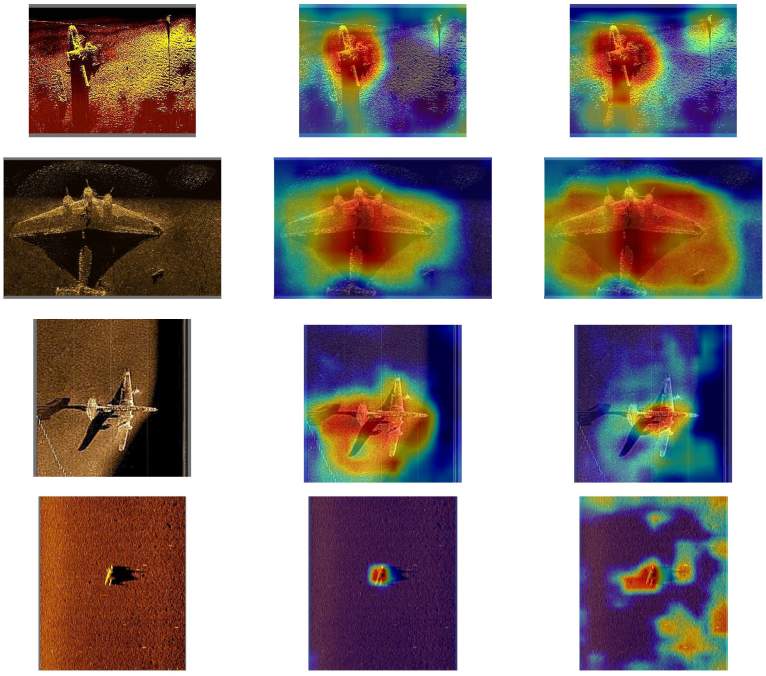
(**a**) (**b**)

Figure 11. Influence of warm-up with cosine annealing and transfer learning on network. (**a**) mAP@0.5 and (**b**) loss.

It is evident that warm-up with cosine annealing and transfer learning accelerate the training process significantly. After the 150th epoch, the mAP@0.5 using warm-up with cosine annealing and transfer learning is stable, which indicates that the model has converged, while the mAP@0.5 of baseline is stable only after the 250th epoch. Warm-up with cosine annealing and transfer learning training strategies also help the network reduce the loss by 0.875% and 0.894%, respectively.

The network trained on HSCTD has improved in P, R, mAP@0.5 and mAP@0.5:0.95, reaching 93.2%, 92.6%, 89.6% and 73.5%, respectively, which are 5%, 3.2%, 4.7% and 2.2% higher than the baseline. This reflects that the simulated images generated by ST has similar characteristics with the real SSS images.

The network inserting a CE feature encoder has the highest improvement in P and R, reaching 95.1% and 93.9%, respectively, which are 6.9% and 4.5% higher than the baseline. For mAP@0.5 and mAP@0.5:0.95, YOLOv5 plus multi-scale images achieves the highest performance, 90.4% and 77.6%, respectively, which is 5.5% and 6.3% higher than the baseline. In order to better show the effect of a CE feature encoder, the heatmap of the sample is visualized to show the important areas concerned by the network, as shown in Figure 12. It can be seen from Figure 12b,c that the network with a CE feature encoder is more focused on the effective features in the images. Small targets are easy to be confused by background because they occupy fewer pixels. The improvement in small target detection lies in the fact that the CE feature encoder assigns different weights to the feature maps of different channels in the same layer, and the increase in the weight of the feature maps containing the target information improves the detection accuracy.



(a) (b) (c)

Figure 12. Detection results with heatmap. (**a**) Original images; (**b**) detection with a CE feature encoder; and (**c**) detection of the original network.

5.3. Network Complexity Analysis

After sparsity regularization training, how many channels to delete needs to be decided. The model is trained with $\lambda = 0.01$ and shows the effect of pruning channels with different percentages. The results are summarized in Figure 13. For each round of sparsity regularization training, pruning and fine-tuning, it only cuts 10% of the channels. So, 20% pruned means two rounds of the above process, etc.

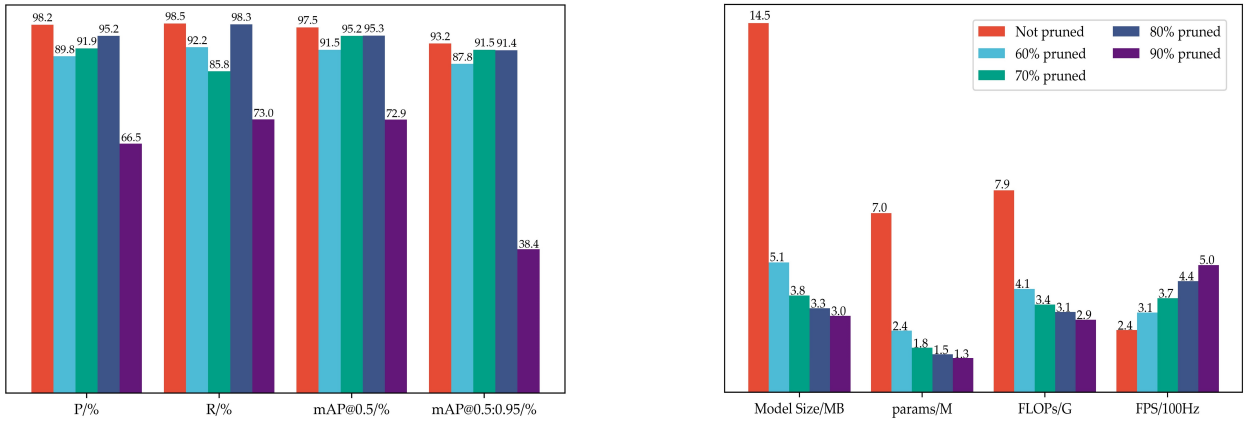


Figure 13. Performance of YOLO-slimming with different pruning ratios. (**a**) Accuracy; (**b**) Complexity.

It can be concluded that only when the pruning ratio is up to 90%, the classification performance can be significantly reduced, as shown in Figure 13a. The complexity of the model decreased significantly, as shown in Figure 13b, where FPS increased from 244 to 501. Compared to the optical domain, the complexity of the underwater scene is relatively low. After channel-level sparsity regularization, approximately 80% of the information processed by the channels in the model is redundant, and this can be removed directly for the purpose of model lightweighting.

5.4. Real-Time Inception

To verify the availability of online real-time detection of YOLO-slimming on SSS, a small embedded GPU NVIDIA Jetson Nano was selected. The hardware performance comparison between Jetson Nano and the laboratory platform NVIDIA GeForce RTX2070 is shown in Table 4.

Table 4. Hardware performance.

Device	Power Dissipation	CUDA Cores
Jetson Nano	10 W	128
RTX 2070	200 W	2304

The CUDA core is the device's information processing unit. The Jetson Nano is only approximately 1/20 of the RTX 2070, and the maximum power dissipation of the Jetson Nano is only 10 W. This is suitable for carrying the Jetson Nano to the SSS for real-time target detection. Three latest light-weight target detection algorithms are selected (i.e., YOLOv5-s [25], YOLOv6-s [36], YOLOv7-tiny [37]) for real-time test. All networks are trained and inferred under the same conditions. Figure 14 shows some detection results of the above four models and more detailed experimental results are listed in Table 5.

Table 5. Real-time test results.

Model	*P*	*R*	mAP@0.5	mAP@0.5:0.95	FPS
YOLO-slimming	98.1%	97.2%	95.3%	91.4%	45
YOLOv5-s	93.2%	92.6%	89.6%	73.5%	12
YOLOv6-s	95.1%	93.9%	95.4%	90.9%	24
YOLOv7-tiny	87.8%	88.1%	96.6%	91.7%	18

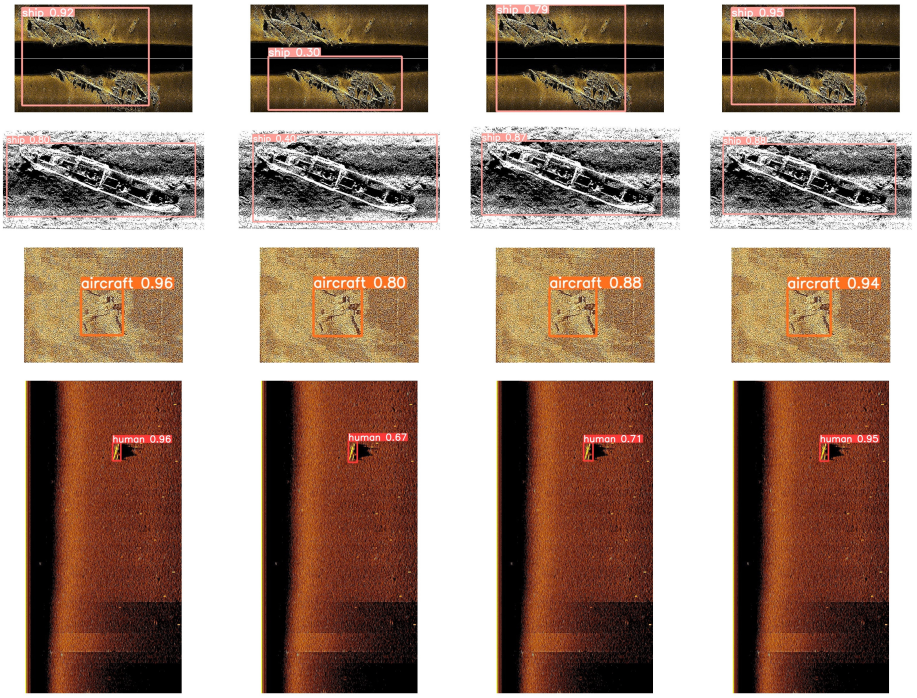


Figure 14. Some representative detection results of four models using Jetson Nano. (**a**) YOLO-slimming; (**b**) YOLOv5-s; (**c**) YOLOv6-s; (**d**) YOLOv7-tiny.

The four models identify fractured, low-definition, eroded and small targets effectively. However, original YOLOv5-s did not completely identify the entire sunken ship, and its other target detection confidence was the lowest. YOLOv6-s and YOLOv7-tiny have excellent performance in the detection of large targets. Additionally, for small ones, the proposed model has the best detection confidence.

YOLO-slimming exceeds YOLOv5-s in all indicators. The mAP@0.5 of YOLOv6-s is 0.1% higher than the proposed network, but its mAP@0.5:0.95 is 0.5% lower. This shows that YOLOv6-s has poor detection results for multi-scale targets, and its backbone cannot effectively retain the information contained in small targets in the feature extraction stage. YOLOv7-tiny is 1.3% and 0.3% higher in mAP@0.5 and mAP@0.5:0.95, respectively, but its inference speed is only half of the proposed network. The running speed of YOLOv5-s and YOLOv6-s is also significantly lower than our model. This is because YOLOv5-s only reduces the number of layers and channels by presetting reduction ratio, and does not cut the invalid channels. YOLOv6-s and YOLOv7-tiny adopt the re-parameterization to accelerate inference of the multi-branch structure, but it is not accurate to the channel-level optimization. YOLO runs at the fastest speed while ensuring accuracy. YOLO-slimming runs at the fastest speed, with FPS reaching 45, which meets the standard of real-time detection (the general requirement is 30), while ensuring accuracy.

6. Conclusions

As a fundamental device for AUVs, real-time autonomous SSS target detection is of high application value. In this paper, YOLO-slimming, a low computational complexity network based on YOLOv5, is proposed, which uses channel-level sparsity regularization to reduce the running time. Meanwhile, efficient feature encoders are added to the Backbone to improve detection accuracy. Experimental results on hybrid SSS images demonstrate the superiority of the model for complex target (e.g., fractured targets) detection accuracy measurements. A comparison of runtimes on a small GPU shows that YOLO-slimming is faster than other competing methods and achieves real-time detection. Based on its efficiency and effectiveness, YOLO-slimming provides a practical solution for real-time, online SSS object detection. In the future, effective fusion of priori information based on sonar domain

knowledge with deep learning models such as Transformer and semantic segmentation will be explored, and acoustic vision-based target localization will be developed on the basis of detection.

Author Contributions: Z.L. was responsible for the whole experiment; D.C. provided ideas for research; T.L.Y. provided suggestions for manuscript writing; J.Z. proofread the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Key R&D Program of China (under grant 2021YFC3001504) and the National Nature Science Foundation of China (under grant 52272424).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in this study.

Data Availability Statement: Microsoft COCO. (2017). Common Objects in Context. Retrieved from https://cocodataset.org/, accessed on 25 January 2022.

Acknowledgments: All authors thank the anonymous reviewers for constructive comments that helped improve this manuscript.

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

References

1. Kaeser, A.J.; Litts, T.L.; Tracy, T.W. Using Low-cost Side-scan Sonar for Benthic Mapping throughout the Lower Flint River, Georgia, USA. *River Res. Appl.* **2013**, *29*, 634–644. [CrossRef]
2. Kennish, M.J.; Haag, S.M.; Sakowicz, G.P.; Tidd, R.A. Side-scan sonar imaging of subtidal benthic habitats in the Mullica River Great Bay estuarine system. *J. Coast. Res.* **2004**, 227–240. [CrossRef]
3. Llorens-Escrich, S.; Tamarit, E.; Hernandis, S.; Sanchez-Carnero, N.; Rodilla, M.; Perez-Arjona, I.; Moszynski, M.; Puig-Pons, V.; Tena-Medialdea, J.; Espinosa, V. Vertical Configuration of a Side Scan Sonar for the Monitoring of Posidonia oceanica Meadows. *J. Mar. Sci. Eng.* **2021**, *9*, 1332. [CrossRef]
4. Wright, R.G.; Baldauf, M. Hydrographic Survey in Remote Regions: Using Vessels of Opportunity Equipped with 3-Dimensional Forward-Looking Sonar. *Mar. Geod.* **2016**, *39*, 439–457. [CrossRef]
5. LeHardy, P.K.; Moore, C. Deep Ocean Search for Malaysia Airlines Flight 370. In Proceedings of the Oceans Conference, St. John's, NB, Canada, 14–19 September 2014.
6. LeHardy, P.K.; Larsen, J. Deepwater Synthetic Aperture Sonar and the Search for MH370. In Proceedings of the OCEANS MTS/IEEE Conference, Washington, DC, USA, 19–22 October 2015.
7. Pailhas, Y.; Petillot, Y.; Brown, K.; Mulgrew, B. Spatially Distributed MIMO Sonar Systems: Principles and Capabilities. *IEEE J. Ocean. Eng.* **2017**, *42*, 738–751. [CrossRef]
8. Yu, H.P.; Li, Z.Y.; Li, D.L.; Shen, T.S. Bottom Detection Method of Side-Scan Sonar Image for AUV Missions. *Complexity* **2020**, *2020*, 9. [CrossRef]
9. Grothues, T.M.; Newhall, A.E.; Lynch, J.F.; Vogel, K.S.; Gawarkiewicz, G.G. High-frequency side-scan sonar fish reconnaissance by autonomous underwater vehicles. *Can. J. Fish. Aquat. Sci.* **2017**, *74*, 240–255. [CrossRef]
10. Batchelor, C.L.; Montelli, A.; Ottesen, D.; Evans, J.; Dowdeswell, E.K.; Christie, F.D.W.; Dowdeswell, J.A. New insights into the formation of submarine glacial landforms from high-resolution Autonomous Underwater Vehicle data. *Geomorphology* **2020**, *370*, 17. [CrossRef]
11. Popli, R.; Kansal, I.; Garg, A.; Goyal, N.; Garg, K. Classification and recognition of online hand-written alphabets using machine learning methods. *IOP Conf. Ser. Mater. Sci. Eng.* **2021**, *1022*, 012111. [CrossRef]
12. Singh, T.P.; Gupta, S.; Garg, M.; Gupta, D.; Alharbi, A.; Alyami, H.; Anand, D.; Ortega-Mansilla, A.; Goyal, N. Visualization of Customized Convolutional Neural Network for Natural Language Recognition. *Sensors* **2022**, *22*, 2881. [CrossRef]
13. Hasija, T.; Kadyan, V.; Guleria, K.; Alharbi, A.; Alyami, H.; Goyal, N. Prosodic Feature-Based Discriminatively Trained Low Resource Speech Recognition System. *Sustainability* **2022**, *14*, 614. [CrossRef]
14. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
15. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
16. Aziz, L.; Salam, M.S.B.; Sheikh, U.U.; Ayub, S. Exploring Deep Learning-Based Architecture, Strategies, Applications and Current Trends in Generic Object Detection: A Comprehensive Review. *IEEE Access* **2020**, *8*, 170461–170495. [CrossRef]
17. Wang, C.-Y.; Bochkovskiy, A.; Liao, H.-Y.M. Scaled-yolov4: Scaling cross stage partial network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 13029–13038.
18. Xu, X.; Zhang, X.; Zhang, T. Lite-yolov5: A lightweight deep learning detector for on-board ship detection in large-scene sentinel-1 sar images. *Remote Sens.* **2022**, *14*, 1018. [CrossRef]

19. Li, G.Q.; Zhang, M.; Zhang, J.W.; Zhang, Q.R. OGCNet: Overlapped group convolution for deep convolutional neural networks. *Knowl.-Based Syst.* **2022**, *253*, 12. [CrossRef]
20. Li, G.Q.; Zhang, J.W.; Zhang, M.; Wu, R.X.; Cao, X.Y.; Liu, W.Z. Efficient depthwise separable convolution accelerator for classification and UAV object detection. *Neurocomputing* **2022**, *490*, 1–16. [CrossRef]
21. Dai, X.; Zhang, P.; Wu, B.; Yin, H.; Sun, F.; Wang, Y.; Dukhan, M.; Hu, Y.; Wu, Y.; Jia, Y. Chamnet: Towards efficient network design through platform-aware model adaptation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 11398–11407.
22. Kazama, Y.; Yamamoto, T. Shallow water bathymetry correction using sea bottom classification with multispectral satellite imagery. In Proceedings of the Conference on Remote Sensing of the Ocean, Sea Ice, Coastal Waters, and Large Water Regions, Warsaw, Poland, 11–12 September 2017.
23. Ruan, F.X.; Dang, L.X.; Ge, Q.; Zhang, Q.; Qiao, B.J.; Zuo, X.Y. Dual-Path Residual "Shrinkage" Network for Side-Scan Sonar Image Classification. *Comput. Intell. Neurosci.* **2022**, *2022*, 6962838. [CrossRef] [PubMed]
24. Cheng, Z.; Huo, G.Y.; Li, H.S. A Multi-Domain Collaborative Transfer Learning Method with Multi-Scale Repeated Attention Mechanism for Underwater Side-Scan Sonar Image Classification. *Remote Sens.* **2022**, *14*, 355. [CrossRef]
25. Song, Y.; He, B.; Liu, P. Real-Time Object Detection for AUVs Using Self-Cascaded Convolutional Neural Networks. *IEEE J. Ocean. Eng.* **2021**, *46*, 56–67. [CrossRef]
26. Yulin, T.; Jin, S.; Bian, G.; Zhang, Y. Shipwreck target recognition in side-scan sonar images by improved YOLOv3 model based on transfer learning. *IEEE Access* **2020**, *8*, 173450–173460. [CrossRef]
27. Aubard, M.; Madureira, A.; Madureira, L.; Pinto, J. Real-Time Automatic Wall Detection and Localization based on Side Scan Sonar Images. In Proceedings of the IEEE/OES Autonomous Underwater Vehicles Symposium (AUV), Singapore, 19–21 September 2022.
28. Li, Y.; Wu, M.Y.; Guo, J.H.; Huang, Y. A Strategy of Subsea Pipeline Identification with Sidescan Sonar based on YOLOV5 Model. In Proceedings of the 21st International Conference on Control, Automation and Systems (ICCAS), Jeju, Republic of Korea, 12–15 October 2021; pp. 500–505.
29. Yu, Y.C.; Zhao, J.H.; Gong, Q.H.; Huang, C.; Zheng, G.; Ma, J.Y. Real-Time Underwater Maritime Object Detection in Side-Scan Sonar Images Based on Transformer-YOLOv5. *Remote Sens.* **2021**, *13*, 3555. [CrossRef]
30. Sun, Y.S.; Zheng, H.T.; Zhang, G.C.; Ren, J.F.; Xu, H.; Xu, C. DP-ViT: A Dual-Path Vision Transformer for Real-Time Sonar Target Detection. *Remote Sens.* **2022**, *14*, 5807. [CrossRef]
31. Yu, F.; He, B.; Liu, J.; Wang, Q. Dual-branch framework: AUV-based target recognition method for marine survey. *Eng. Appl. Artif. Intell.* **2022**, *115*, 105291. [CrossRef]
32. Yin, P.H.; Zhang, S.; Qi, Y.Y.; Xin, J. Quantization and Training of Low Bit-width Convolutional Neural Networks for Object Detection. *J. Comput. Math.* **2019**, *37*, 349–360. [CrossRef]
33. Kim, Y.; Choi, O.; Hwang, W. Low bit-based convolutional neural network for one-class object detection. *Electron. Lett.* **2021**, *57*, 255–257. [CrossRef]
34. Wu, J.; Zhu, J.H.; Tong, X.; Zhu, T.L.; Li, T.Y.; Wang, C.Z. Dynamic activation and enhanced image contour features for object detection. *Connect. Sci.* **2022**. [CrossRef]
35. Yu, K.; Cheng, Y.F.; Tian, Z.T.; Zhang, K.H. High Speed and Precision Underwater Biological Detection Based on the Improved YOLOV4-Tiny Algorithm. *J. Mar. Sci. Eng.* **2022**, *10*, 1821. [CrossRef]
36. Li, C.; Li, L.; Jiang, H.; Weng, K.; Geng, Y.; Li, L.; Ke, Z.; Li, Q.; Cheng, M.; Nie, W. YOLOv6: A single-stage object detection framework for industrial applications. *arXiv* **2022**, arXiv:2209.02976.
37. 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. *arXiv* **2022**, arXiv:2207.02696.
38. Wang, C.-Y.; Liao, H.-Y.M.; Wu, Y.-H.; Chen, P.-Y.; Hsieh, J.-W.; Yeh, I.-H. CSPNet: A new backbone that can enhance learning capability of CNN. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020; pp. 390–391.
39. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [CrossRef]
40. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8759–8768.
41. Zheng, Z.H.; Wang, P.; Ren, D.W.; Liu, W.; Ye, R.G.; Hu, Q.H.; Zuo, W.M. Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation. *IEEE Trans. Cybern.* **2022**, *52*, 8574–8586. [CrossRef]
42. Wang, Q.; Wu, B.; Zhu, P.; Li, P.; Zuo, W.; Hu, Q. ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 11531–11539.
43. Liu, B.; Wang, M.; Foroosh, H.; Tappen, M.; Pensky, M. Sparse convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 806–814.
44. Denos, K.; Ravaut, M.; Fagette, A.; Lim, H.S. Deep Learning applied to Underwater Mine Warfare. In Proceedings of the Oceans Aberdeen Conference, Aberdeen, UK, 19–22 June 2017.

45. Li, R.; Wu, C.-H.; Liu, S.; Wang, J.; Wang, G.; Liu, G.; Zeng, B. SDP-GAN: Saliency detail preservation generative adversarial networks for high perceptual quality style transfer. *IEEE Trans. Image Process.* **2020**, *30*, 374–385. [CrossRef]
46. Omiotek, Z.; Kotyra, A. Flame image processing and classification using a pre-trained VGG16 model in combustion diagnosis. *Sensors* **2021**, *21*, 500. [CrossRef]
47. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
48. Iandola, F.; Moskewicz, M.; Karayev, S.; Girshick, R.; Darrell, T.; Keutzer, K. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv* **2014**, arXiv:1404.1869.
49. Zhou, Y.; Chen, S.; Wu, K.; Ning, M.; Chen, H.; Zhang, P. SCTD1. 0: Sonar common target detection dataset. *Comput. Sci.* **2021**, *48*, 334–339.
50. Xia, Z.Y.; Kim, J. Mixed spatial pyramid pooling for semantic segmentation. *Appl. Soft. Comput.* **2020**, *91*, 9. [CrossRef]
51. Johnson, J.M.; Khoshgoftaar, T.M. Survey on deep learning with class imbalance. *J. Big Data* **2019**, *6*, 27. [CrossRef]
52. Xu, Y.; Wang, H.; Liu, X. An improved multi-branch residual network based on random multiplier and adaptive cosine learning rate method. *J. Vis. Commun. Image Represent.* **2019**, *59*, 363–370. [CrossRef]
53. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks? In Proceedings of the 28th Conference on Neural Information Processing Systems (NIPS), Montreal, QC, Canada, 8–13 December 2014.
54. Arifuzzaman, M.; Arslan, E. Learning Transfers via Transfer Learning. In Proceedings of the 8th IEEE Workshop on Innovating the Network for Data-Intensive Science (INDIS), St. Louis, MO, USA, 14–19 November 2021; pp. 34–43.
55. Zheng, L.; Shen, L.; Tian, L.; Wang, S.; Wang, J.; Tian, Q. Scalable Person Re-identification: A Benchmark. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1116–1124.