





# Light-YOLOv4: An Edge-Device Oriented Target Detection Method for Remote Sensing Images

Xiaojie Ma , Kefeng Ji , *Member, IEEE*, Boli Xiong, Linbin Zhang , Sijia Feng ,  
and Gangyao Kuang, *Senior Member, IEEE*

**Abstract**—Most deep-learning-based target detection methods have high computational complexity and memory consumption, and they are difficult to deploy on edge devices with limited computing resources and memory. To tackle this problem, this article proposes to learn a lightweight detector named Light-YOLOv4, and it is obtained from YOLOv4 through model compression. To this end, first, we perform sparsity training by applying L1 regularization to the channel scaling factors, so the less important channels and layers can be found. Second, channel pruning and layer pruning are enforced on the network to prune the less important parts, which could significantly reduce network's width and depth. Third, the pruned model is retrained with a knowledge distillation method to improve the detection accuracy. Fourth, the model is quantized from FP32 to FP16, and it could further accelerate the model with almost no loss of detection accuracy. Finally, to evaluate Light-YOLOv4's performance on edge devices, Light-YOLOv4 is deployed on NVIDIA Jetson TX2. Experiments on the SAR ship detection dataset (SSDD) demonstrate that the model size, parameter size, and FLOPs of Light-YOLOv4 have been reduced by 98.63%, 98.66%, and 91.30% compared with YOLOv4, and the detection speed has been increased to  $4.2\times$ . While the detection accuracy of Light-YOLOv4 is only slightly reduced, for example, the mAP has only reduced by 0.013. Besides, experiments on the Gaofen Airplane dataset also prove the feasibility of Light-YOLOv4. Moreover, compared with other state-of-the-art methods, such as SSD and FPN, Light-YOLOv4 is more suitable for edge devices.

**Index Terms**—Edge device, model compression, NVIDIA Jetson TX2, remote sensing, target detection, YOLOv4.

## I. INTRODUCTION

WITH the development of remote sensing technology, the era of remote sensing big data has arrived [1], [2]. The traditional target detection process, that is, transmitting remote sensing data to the ground station and then performing target detection, has been difficult to meet the real-time requirements [3]. In this context, applying edge computing technology to the field of remote sensing and deploying remote sensing image target detection on edge devices such as on-orbit satellites or UAVs

can undoubtedly save a lot of time and improve the response speed [4], [5]. Therefore, how to design fast and efficient target detection algorithms suitable for edge devices has attracted more and more attention in the field of remote sensing.

Recently, deep learning has been widely used in the field of target detection with its powerful generalization ability [6], [58]. Different from the traditional methods that require manually designed features, these methods could extract features automatically and realize end-to-end target detection [7], [8]. Even more, the detection performance of deep learning-based methods is much better. In general, target detection methods based on deep learning can be divided into two categories [9], [10]. One is the single-stage detector, such as YOLO series [11]–[14], SSD [15], and RetinaNet [16], which have lower computational complexity and faster detection speed. The other is the two-stage detector, such as Faster RCNN [17] and feature pyramid network (FPN) [18], which have the better detection accuracy. Some scholars have improved these methods and applied them to target detection in remote sensing images [19], [20]. For example, for multiscale target detection, Dong *et al.* [21] improved Faster RCNN by replacing traditional nonmaximum suppression (NMS) with Sig-NMS in the stage of region proposal network, which could significantly reduce the possibility of missing small targets. Cui *et al.* [22] proposed a detection method based on a dense attention pyramid network (DAPN). By extracting abundant features, including resolution and semantic information, the detection performance of multiscale ship targets is improved. For multioriented target detection, An *et al.* [23] improved DRBox-v1 using FPN, focal loss, and a modified encoding scheme, and then proposed a DRBox-v2 detector, which could detect ships in any direction. Li *et al.* [24] proposed a rotatable region-based residual network (R3-Net) to detect multioriented vehicles in remote sensing images and videos, and it has high robustness and detection accuracy. For dense target detection, Wang *et al.* [25] added spatial groupwise enhance (SGE) attention module to CenterNet, and experimental results indicate that it could detect dense docked ships well. The above methods have made satisfactory progress in terms of detection accuracy. However, they are still computationally expensive and time-consuming, which makes them unsuitable for deployment on edge devices with limited computing resources and memory. Therefore, it is necessary to design lightweight target detection models for remote sensing images.

Lightweight model design methods can generally be divided into two categories [26]. The first category is to design a

Manuscript received July 7, 2021; revised September 5, 2021 and September 25, 2021; accepted October 9, 2021. Date of publication October 14, 2021; date of current version November 3, 2021. This work was supported by the National Natural Science Foundation of China under Grant 62001480. (*Corresponding author: Kefeng Ji.*)

The authors are with the State Key Laboratory of Complex Electromagnetic Environment Effects on Electronics and Information System, College of Electronic Science, National University of Defense Technology, Changsha 410073, China (e-mail: mxj286@foxmail.com; jikefeng@nudt.edu.cn; bolixiong@qq.com; zlbndt@163.com; fengsijia12@nudt.edu.cn; kuangyeats@hotmail.com).

Digital Object Identifier 10.1109/JSTARS.2021.3120009

lightweight model before training, including redesign the backbone network structure or convolution calculation unit (e.g., group convolution [27] and depthwise separable convolution [28]), for example, Li *et al.* [29] improved the feature extraction basic network (backbone network) structure and recognition and positioning task network of Faster RCNN, then a lightweight ship detector named Lite Faster RCNN was proposed. The detection speed of Lite Faster RCNN is increased by eight times. Zhao *et al.* [30] introduced a depthwise separable convolutional strategy to the network and built a lightweight detector. Ding *et al.* [31] proposed the use of a fully convolutional neural network instead of the fully connected layers in Faster RCNN to detect targets in optical remote sensing images; after that, the memory requirements and time consumption are reduced. Zhou *et al.* [32] combined the idea of dense connections, residual connections, and group convolution and proposed the Lira-YOLO ship detector. Compared with tiny-YOLOv3 [33], Lira-YOLO has a higher detection accuracy and lower computational complexity. These methods can meet the requirements of edge devices, but they rely on the manual prior knowledge too much. Therefore, the second type of lightweight model design method, namely model compression, is adopted in this article.

Three model compression methods have been widely used in recent years, including model pruning, knowledge distillation (KD), and quantization [34]. Model pruning aims to remove the redundant channels or layers of the network, which could significantly reduce the parameter size and model size [35], [36]. There are different pruning strategies. For example, Liu *et al.* proposed to use sparse training to judge the importance of each channel in the network and then prune the less important channels [35]. Guo *et al.* proposed progressive channel pruning to accelerate CNN, which could iteratively prune a small number of channels from several selected layers [36]. In this article, we have improved the methods in [35] by increasing the layer pruning, so both the depth and width of the network can be reduced. Second, KD takes the large and trained network as the teacher network; then the larger network is used to guide the training of the small student network [37], [38]. Compared with other training methods, the detection accuracy of the network trained by KD is much higher. Besides, the student network can be a well-designed lightweight network or a pruned network. At last, the core idea of quantization is to compress the network by reducing the number of bits of each weight [39], [40], for instance, from 32-bit floating point to 16-bit floating point or 8-bit integer. After quantization, the model size and detection speed are greatly reduced. In practice, these methods are often used in combination to get better performance. For example, Zhang *et al.* applied the structured pruning method to compress the network, and then the KD is employed to improve the recognition accuracy of the compressed network [54]. Chen *et al.* combined squeezing, deep compression, and a novel fast computation algorithm to compress and accelerate the network [55]. These two methods have achieved good results for resource-constrained SAR target recognition, and they are also helpful for the compression of the target detection network.

Based on the above ideas, this article proposes an edge-device-oriented target detection method, and the detector is compressed

from YOLOv4 [14]. Compared with YOLOv3 [13], YOLOv4 has made a series of improvements, which could improve the detection accuracy at a low cost. In detail, first, sparsity training is performed on the trained YOLOv4, and then channel pruning and layer pruning are conducted to get the slimmer and shallower pruned network, respectively. Second, the KD scheme is used to retrain the pruned network, and YOLOv4 is the teacher network during this process. KD makes the pruned network have a better detection performance. The next step is quantization; it converts the weight of the pruned network to 16-bit floating point (FP16) from 32-bit floating point (FP32). Finally, the compressed model is implemented on NVIDIA Jetson TX2, which is an edge device with excellent performance and low power consumption (less than 15 W) [42]. Experiments on the SSDD dataset [43] and Gaofen Airplane dataset [57] demonstrate that the proposed method is more efficient and faster than the existing methods, and thus, it is more suitable for edge devices.

The main contributions of this article are indicated as follows.

- 1) An edge-device-oriented target detection method was proposed. To get it, a series of measures, including sparsity training, channel and layer pruning, KD, and quantization, are used to compress YOLOv4.
- 2) Our proposed model was successfully deployed on the NVIDIA Jetson TX2 embedded device, which proved the feasibility of our proposed model on edge devices.
- 3) Experiments on SSDD and Gaofen Airplane datasets indicate that our proposed method can achieve competitive detection accuracy with fewer FLOPS and higher speed, which could meet the requirement of edge devices with limited computing capability and memory.

The rest of this article is organized as follows. Section II briefly introduces the basics of YOLOv4. Section III introduces our proposed target detection method. In Section IV, the detectors are deployed on NVIDIA Jetson TX2, what is more, the experimental results and discussions on SSDD and Gaofen Airplane datasets are exhibited. Finally, Section V concludes this article.

## II. BASIC OF YOLOV4

YOLO [11] is one of the most representative one-stage target detection algorithms. So far, YOLO has been updated for four versions, and each version brings great performance improvements. YOLOv2 [12] removed the full connection layer in YOLO. What is more, it introduced anchor boxes to predict the bounding boxes. YOLOv3 replaced the backbone of YOLOv2 with Darknet53 and adopted multiscale prediction strategy [13], which significantly improved the detection speed and accuracy.

YOLOv4 is a further improvement of the YOLO series algorithms. It is based on YOLOv3 and has merged in many excellent detection tricks to improve detection accuracy. The network architecture of YOLOv4 is shown in Fig. 1.

Compared with YOLOv3, YOLOv4 mainly made the following improvements. First, YOLOv4 introduced the cross-stage partial (CSP) [44] structure to the backbone and proposed CSPDarkNet53 [14], which maintains the detection accuracy while reducing the weight of the backbone network. Second,

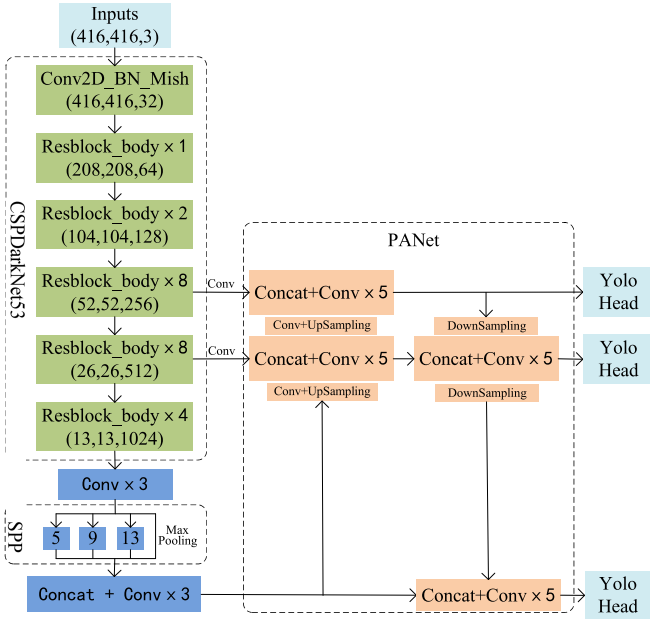


Fig. 1. Network architecture of YOLOv4.

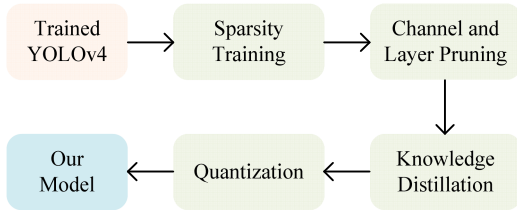


Fig. 2. Overall architecture of our proposed method.

FPN and path aggregation network [45] were introduced to YOLOv4, and they could make full use of context information to improve the feature extraction ability. Third, it is proposed to use the mosaic method to preprocess the training data, which can improve the detection ability for small-scale targets. What is more, Mish activation function, SAM-block, and CIoU were used in YOLOv4, which contribute a lot to the detection accuracy [46].

Recently, YOLOv4 has been widely used in the field of remote sensing and achieved good results [47]. Based on these considerations, this article chooses YOLOv4 as the baseline.

### III. PROPOSED METHOD

The overall architecture of our proposed method is presented in Fig. 2. The well-trained YOLOv4 is selected as the baseline; then we continue to conduct sparsity training, channel and layer pruning, KD, and quantization on it to get our model. Among them, sparsity training is used to find the less important channels and layers, channel and layer pruning are used to prune the model, KD could improve the detection accuracy of the pruned model, and quantization could further improve the detection speed. The details of each comment will be introduced in the following parts.

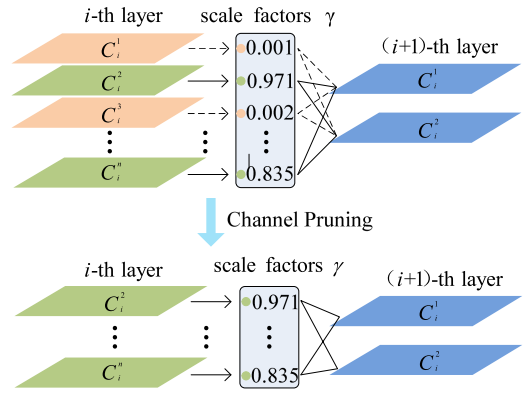


Fig. 3. Schematic diagram of channel pruning.

#### A. Sparsity Training

The purpose of sparsity training is to evaluate the importance of each channel or layer of the model. In YOLOv4, each convolutional layer is followed by a BatchNorm (BN) layer, which could improve the generalization and convergence speed. BN layer makes use of the min-batch statistics to normalize the convolutional features, and it can be represented by the following equation:

$$\hat{x} = \frac{x_{in} - \bar{x}}{\sqrt{\sigma^2 + \varepsilon}}, \quad x_{out} = \gamma \cdot \hat{x} + \beta \quad (1)$$

where  $\bar{x}$  and  $\sigma^2$  represent the mean and variance of the input features in a mini-batch, while  $\gamma$  and  $\beta$  are the trainable scaling factor and shift factor, respectively. Among these parameters, the trainable scaling factor  $\gamma$  can be used to measure the importance of its corresponding channel. It means that the smaller  $\gamma$  is, the less important the corresponding channel is.

In the process of sparsity training, L1 regularization is performed on the scaling factor  $\gamma$  to find the unimportant channels. The loss function of sparsity training is represented by

$$L = \text{loss}_{\text{YOLOv4}} + \alpha \sum_{\gamma \in \Gamma} \|\gamma\|_1 \quad (2)$$

where  $\text{loss}_{\text{YOLOv4}}$  is the total loss of YOLOv4, and  $\|\gamma\|_1$  denotes L1-norm.  $\alpha$  is a hyperparameter, which is used to balance the two parts of loss function.

#### B. Channel Pruning and Layer Pruning

After sparsity training, most of the scaling factors are close to 0, and their corresponding channels contribute little to the detection accuracy. Therefore, they can be pruned without losing too much detection accuracy.

The first thing to do is channel pruning, as shown in Fig. 3. Channel pruning could significantly reduce the width of the model. To this end, a global pruning ratio  $\delta$  is introduced to determine which channel should be pruned. In detail, sort all the scaling factors  $\gamma$  from small to large to get the sequence  $A$  and  $\hat{\gamma}$  corresponds to the  $\delta$ th percentile value in the sequence. What is more, a local threshold  $\varepsilon$  is introduced to prevent overpruning on a specific layer, where  $\varepsilon$  is the  $k$ th percentile of all  $\gamma$  in the

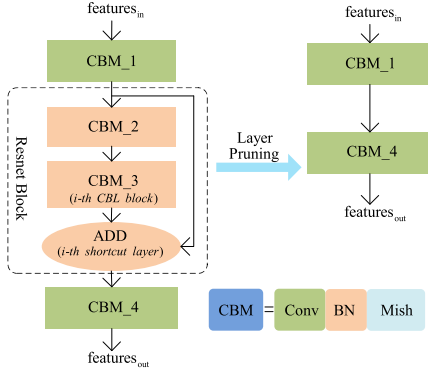


Fig. 4. Schematic diagram of layer pruning.

layer. After that, the channel whose scaling factor is less than the minimum of  $\hat{\gamma}$  and  $\varepsilon$  will be cut off.

It is worth noting that the residual block in the network needs to be treated carefully. First, according to the global threshold  $\hat{\gamma}$  and local threshold  $\varepsilon$ , a pruning mask for all convolutional layers is constructed. Second, the layers that have connections with the shortcut layer should have the same channel number. To match it, we iterate through the pruning masks of all connected layers and perform OR operations on these pruning masks to obtain the final pruning mask [48].

Layer pruning could reduce the depth of the model, and it is only implemented on the shortcut layers of the backbone network. Similar to channel pruning, layer pruning also determines which shortcut layers to be pruned according to the scaling factors. As shown in Fig. 4, we define the first CBM block before the  $i$ th shortcut layer as the  $i$ th CBM block. Suppose that the  $i$ th CBM block has  $N$  channels ( $N$  scaling factors), and the mean  $\bar{\gamma}_i$  of these scaling factors can be represented by

$$\bar{\gamma}_i = \frac{1}{N} \sum_{j=1}^N \gamma_i^j, i = 1, 2, \dots, 23 \quad (3)$$

where  $\gamma_i^j$  is the  $j$ th scaling factor of the  $i$ th CBM block.

Calculate the mean  $\bar{\gamma}_i$  ( $i = 1, 2, \dots, 23$ ), then select the minimum  $M$  values, and their corresponding shortcut layers will be pruned. As shown in Fig. 4, in order to maintain the integrity of the YOLOv4 network structure, the shortcut layer and the two CBM blocks in front of it will be cut off at the same time. It means that a total of  $3 \times M$  layers will be pruned in the layer pruning stage.

### C. Knowledge Distillation

After channel pruning and layer pruning, the width and depth of the model are significantly reduced, and we get a lighter detector. However, model pruning will lead to the loss of detection accuracy. Here, the strategy of KD is used to retrain the pruned model to improve the detection performance.

KD was proposed to transfer the knowledge from a large teacher network to a small student network. In this article, YOLOv4 is used as the teacher network, and the pruned network

is the student network. Only the backbone network is pruned in the previous step; therefore, the teacher network and student network have the same detection framework. As shown in Fig. 5, the images are put into the teacher network and student network, and different outputs are produced in the feature extraction layer (hint), classification result (classification), and regression result (regression). By comparing the outputs of two networks, the distillation loss can be constructed.

In general, the distillation loss is called soft loss, and the detection loss is called hard loss, and they constitute the total loss. The calculation process of total loss can be represented by the following equation:

$$L_{\text{total\_loss}} = \delta \times L_{\text{soft}}(s, t) + (1 - \delta) \times L_{\text{hard}}(s, T). \quad (4)$$

In (4),  $L_{\text{soft}}(s, t)$  is the distillation loss, and  $L_{\text{hard}}(s, T)$  is the detection loss.  $s$  and  $t$  are the student network and teacher network, respectively.  $T$  is the ground truth label.  $\delta \in [0, 1]$  is a hyperparameter to balance the distillation loss and the detection loss through the backpropagation algorithm to update the weight of the student network, which could reduce the loss and make the output of the student network gradually approach the teacher network.

In detail,  $L_{\text{hard}}(s, T)$  is the loss of YOLOv4, and it can be expressed by

$$\begin{aligned} L_{\text{hard}} &= L_{\text{loc}}(s, T) + L_{\text{conf}}(s, T) + L_{\text{cls}}(s, T) \\ &= L_{\text{loc}}(x_s, x_T, y_s, y_T, w_s, w_T, h_s, h_T) \\ &\quad + L_{\text{conf}}(p_s, p_T) + L_{\text{cls}}(p_s, p_T) \end{aligned} \quad (5)$$

where  $L_{\text{loc}}(s, T)$ ,  $L_{\text{conf}}(s, T)$ , and  $L_{\text{cls}}(s, T)$  are the bounding-box loss, confidence loss, and classification loss.  $L_{\text{soft}}(s, t)$  can be represented by

$$L_{\text{soft}} = L_{\text{hint}}(s, t) + L_{\text{loc}}(s, t) + L_{\text{conf}}(s, t) + L_{\text{cls}}(s, t). \quad (6)$$

In (6),  $L_{\text{hint}}(s, t)$  is the hint loss, which is used to measure the difference between the feature maps output by the student network and the teacher network. Here we use Euclidean distance to calculate  $L_{\text{hint}}(s, t)$ .

In summary, in the process of KD, the student network makes the output of the teacher network as the ‘‘soft target.’’ Compared with the ground truth label, the ‘‘soft target’’ contains more information, and it is conducive to the training of the student network [37]. On the other hand, the prediction results of the teacher network are not all correct, and the use of a ground truth label could effectively reduce the possibility of transmitting the wrong information to the student network. Therefore, the combined use of soft loss and hard loss can achieve better performance.

### D. Quantization

Model quantization could effectively reduce the requirements for storage space and memory bandwidth. What is more, it can improve system throughput. In this article, post-training quantization [49] method is taken to quantify the weights, which avoids the time-consuming quantization training or retraining process.



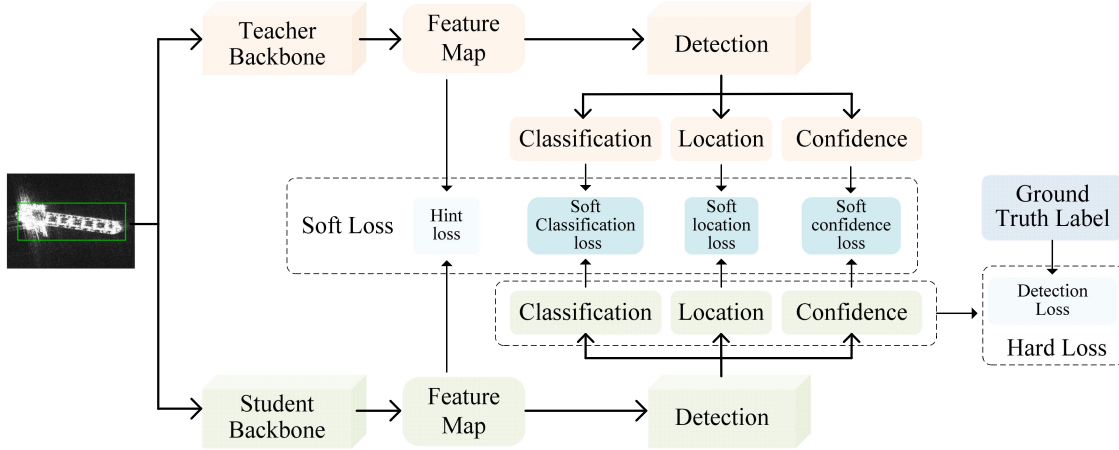


Fig. 5. YOLOv4 detection model KD architecture framework.

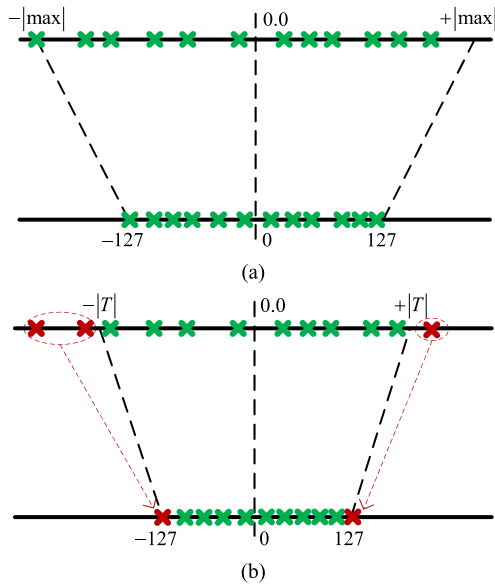


Fig. 6. Schematic diagram of quantization. (a) Maxmin Quantization. (b) Our Quantization.

The most commonly used quantization is linear quantization, and it can be expressed by

$$T = sf * t. \quad (7)$$

In (7),  $sf$  is the scaling factor, and  $T$  and  $t$  are the values before and after quantization, respectively. Obviously, the scaling factor is crucial to quantization. As shown in Fig. 6(a), the traditional linear quantization uses the Maxmin quantization method to determine the quantization threshold, so as to determine the scaling factor. Therefore, the scaling factor is  $sf = \frac{|\max|}{2^{N-1}}$ , where  $N$  is the bit width after quantization. When the data are asymmetric, as shown in Fig. 6(a), there will be a significant accuracy loss.

To solve the problem caused by symmetric data, first, we produce lots of quantization thresholds, which correspond to different quantization results. Second, KL divergence is used to find the quantization scheme with the smallest distribution

difference between the original data and various quantized data (for more details, refer to [56]). KL divergence is often used to measure the difference between two distributions  $p$  and  $q$ , which can be expressed by

$$\text{KL\_divergence}(p||q) = \sum_{i=1}^N p(x_i) \log \frac{p(x_i)}{q(x_i)}. \quad (8)$$

The process of using KL divergence to determine the threshold is as follows.

- 1) For each layer, collect the histograms of activations.
- 2) Generate many quantized distributions with different saturation thresholds.
- 3) Select the best threshold  $T$ , which could minimize  $\text{KL\_divergence}(\text{ref\_distr}||\text{quant\_distr})$ .

It should be noted that compared with the Maxmin quantization method, the threshold  $T$  calculated by KL divergence is smaller, so there will be some values that the range of  $[-T, +T]$ . This part of data needs to be mapped to the boundary after quantization, which is shown in Fig. 6(b). The final scaling factor is  $sf = \frac{|T|}{2^{N-1}}$ .

In fact, the quantization method introduced in this article has been integrated into TensorRT, so we use TensorRT to quantify the model. Besides, TensorRT could optimize the network structure by fusing the three layers (Conv, BN, and Mish) into one layer, which could further improve the detection speed [52], [53].

In summary, channel pruning and layer pruning belong to structured pruning, which can significantly reduce the width and depth of the network, and the pruned models have no special requirements for the platform to achieve acceleration. KD can help improve the detection accuracy of the pruned models. Quantization can significantly reduce the amount of calculation, and it can be applied to most networks. However, the quantized model requires hardware support to complete acceleration. In this article, these methods are combined to compress YOLOv4 to achieve better performance.

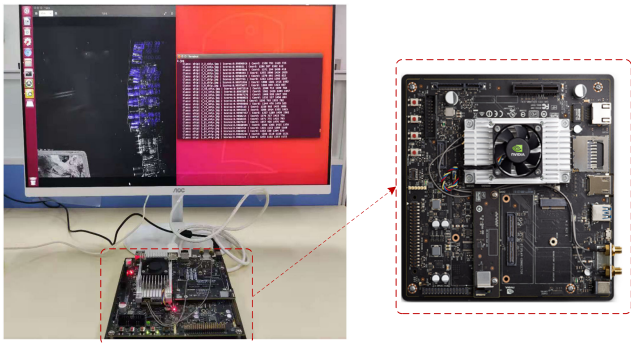


Fig. 7. NVIDIA Jetson TX2 and our experimental devices. The left side shows our experiment devices. The right side of the image shows NVIDIA Jetson TX2.

TABLE I  
MAIN PARAMETERS OF NVIDIA JETSON TX2

| Component | Detailed Information                   |
|-----------|--|
| GPU       | NVIDIA Pascal GPU (256 CUDA cores)     |
| CPU       | Denver 2 CPU (Dual-Core) + ARM A57 CPU |
| Memory    | 8GB 128-bit LPDDR4                     |
| Storage   | 32GB eMMC                              |

#### IV. EXPERIMENTS

In this section, several experiments are implemented to verify the effectiveness of our proposed method. First, the implementations, including experimental platform, evaluation metrics, SSDD, and Gaofen Airplane datasets [57], will be introduced in detail. Next, the ablation experiments will be performed on the SSDD dataset to verify the effectiveness of sparsity training, channel and layers pruning, KD, and quantization. Third, to verify the detection speed and accuracy of our proposed methods, we compare them with several state-of-the-art methods on the SSDD dataset. Finally, experiments are carried out on the Gaofen Airplane dataset, which could prove the feasibility of our proposed method in optical remote sensing target detection.

##### Implementations

1) *Experimental Platform*: The experiments are implemented on Darknet and Pytorch 1.14 framework. We train and compress the models on the computer with Intel Core i7-8700K CPU, NVIDIA GTX 1080Ti GPU, and 16 GB memory. After that, all the models were deployed on NVIDIA Jetson TX2. TX2 is an edge device with low power consumption (less than 15 W) and high performance (about 1T times floating-point operations per second). The NVIDIA Jetson TX2 and our experimental devices are shown in Fig. 7, and its main parameters are presented in Table I. What is more, the operation systems of the TX2 and computer are both Ubuntu 18.04 with CUDA 10.0 and CUDNN 7.6.

2) *Datasets*: SSDD dataset is widely used in SAR ship detection. The images in SSDD [42] dataset mainly come from RadarSat-2, Sentinel-1, and TerraSAR-X sensors, and they are imaged in four polarization modes: HH, VV, HV, and VH. As shown in Fig. 8, these images contain ship targets in both

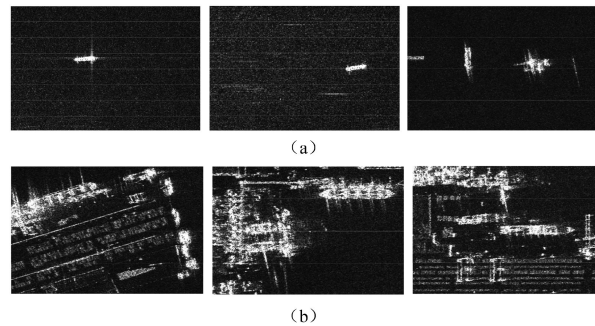


Fig. 8. Ships in offshore areas and nearshore areas in the SSDD dataset. (a) Offshore areas. (b) Nearshore areas.

offshore areas and nearshore areas, and the resolution of them ranges from 1 to 15 m. Comparatively speaking, it is more difficult to detect ships in the nearshore areas due to its complex background. What is more, the size of ships in the SSDD dataset ranges from  $7 \times 7$  to  $211 \times 298$ , and it is widely used to evaluate the performance of multiscale SAR ship detection [50], [51].

There are in total 1160 images and 2456 multiscale ships in the SSDD dataset. We randomly divided these images into three parts: training, validation, and test sets according to the ratio of 7:1:2. What is more, the methods of flipping, mirroring, and changing the contrast are used to augment the images in the train set. After augmentation, there are 6496 images in the training set.

Gaofen Airplane dataset is used for the detection and recognition of optical airplanes, which is released in the 2020 Gaofen challenge. There are 1000 images and 5609 airplanes in this dataset. The size of these images is  $1024 \times 1024$ , and the resolution of them ranges from 0.5 m to 0.8 m. In fact, this dataset subdivides the airplanes into ten categories, but in our experiment, we only perform the detection task (without recognition). What is more, we divide the dataset in a ratio of 7:1:2, and the same methods are used to augment the training set.

3) *Evaluation Metrics*: In this article, a total of eight metrics are used to evaluate the performance of these methods in the experiment, including the recall, precision, F1 score, mean of average precision (mAP), model size, parameter size, inference speed as frames per second (FPS), and floating point of operations (FLOPs).

Recall indicates the proportion of right detected targets in the ground truths, while precision represents the proportion of right detected targets in the detected objects. They can be calculated according to the following two equations.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (9)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (10)$$

In the above two equations, TP (true positive), FP (false positive), and FN (false negative) could represent the number of correctly detected targets, false alarm, and missed detected

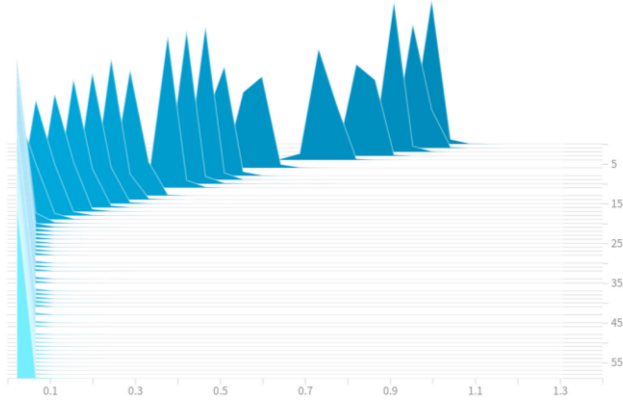


Fig. 9. Distribution of scaling factors during sparsity training. The horizontal axis in the figure represents the absolute value of scaling factors. The left vertical axis represents the distribution probability, and the right vertical axis represents the epoch of sparsity training.

targets, respectively.

$$F1 = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}. \quad (11)$$

Recall and precision contradict each other due to the problem of threshold, and they have limitations. While F1 score (F1) and mAP could evaluate the overall performance of the detector. F1 can be represented by (11).

mAP is defined as (12). It indicates the average area under precision–recall curve, where  $P$  and  $R$  represent the precision and recall of a point on the curve.

$$\text{mAP} = \int_0^1 P(R)dR. \quad (12)$$

All the above four metrics are used to evaluate the accuracy of the detection results. The model size and parameter size indicate the size of the detector. At the same time, FLOPs and FPS can respectively describe the computation complexity and detection speed of the detector.

### Ablation Experiments

In order to verify the effectiveness of each component, several ablation experiments are performed on the SSDD dataset. Two models YOLOv4 and YOLOv4-tiny are implemented as the baseline models. YOLOv4-tiny is a simplified version of YOLOv4, and it is much faster but less accurate. We continue to perform sparsity training, channels and layers pruning, KD, quantization, and TensorRT acceleration on the trained YOLOv4 model. What is more, to investigate the performance of these models in resource-limited environments, all models were deployed on NVIDIA Jetson TX2 device.

1) *Effect of Sparsity Training:* In the process of sparsity training, we count the distribution of all scaling factors (absolute value) in the backbone of YOLOv4 and then visualize the results in Fig. 9. It can be seen that in the beginning, the scaling factors roughly follow the normal distribution with the mean of 1. During sparsity training, the number of smaller scaling factors gradually increases, while the number of larger factors keeps

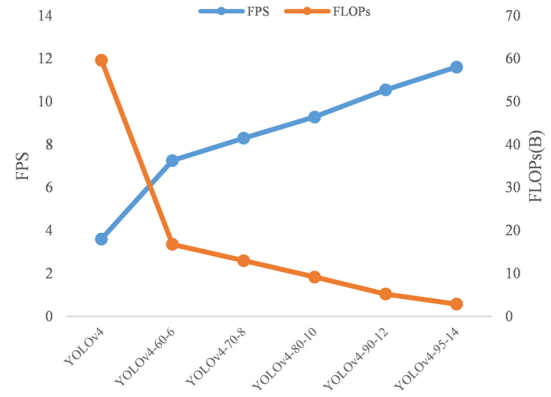


Fig. 10. Comparison of FLOPs and FPS of YOLOv4 and five pruned models.

decreasing. Finally, most of the scaling factors are close to 0. In fact, the channel corresponding to the smaller scaling factor contributes less to the detection accuracy, while the channel corresponding to the larger scaling factor contributes a lot to the detection accuracy.

The value of  $\alpha$  is very important. When the value of  $\alpha$  is too small, the scaling factors are hard to become sparse; conversely, the scaling factors decay so aggressively that the model is easy to be underfitting. We set  $\alpha$  to 0.001 through multiple experiments, and the two parts of the loss function can be well balanced at this time.

2) *Effect of Channel Pruning and Layer Pruning:* After sparsity training, it is necessary to determine the proportion of pruned channels and the number of pruned layers. If there are too few channels and layers to be pruned, the saving of computing resources is limited; on the contrary, the detection accuracy of the model may be difficult to recover. In this experiment, we make the following five pruning schemes:

- 1) pruning 60% channels and six shortcut layers;
- 2) pruning 70% channels and eight shortcut layers;
- 3) pruning 80% channels and ten shortcut layers;
- 4) pruning 90% channels and 12 shortcut layers; and
- 5) pruning 95% channels and 14 shortcut layers.

The pruned models are named YOLOv4-60-6, YOLOv4-70-8, YOLOv4-80-10, YOLOv4-90-12, and YOLOv4-95-14. What is more, the detection accuracy of the pruned models will be much worse, so we use the fine-tuning method to retrain these models to improve their detection performance.

The evaluation metrics of several models are given in Table II. Obviously, compared with YOLOv4, the parameters' size of five pruned models is reduced by 82.12%, 90.19%, 95.52%, 98.66%, and 99.55%, respectively. The model size has been reduced by 83.55%, 90.16%, 95.51%, 98.63%, and 99.53%. FLOPs are reduced by 71.90%, 78.20%, 84.64%, 91.30%, and 95.18%. FPS has become 2.01, 2.30, 2.58, 2.93, and 3.22 times. The comparison of FLOPs and FPS between YOLOv4 and five pruned models is shown in Fig. 10. After fine-tuning, the recall, precision, F1, and mAP of five pruned models have little loss as compared with YOLOv4. The five pruned models are better than YOLOv4-tiny in terms of detection accuracy.



TABLE II  
EVALUATION RESULTS OF THE TWO BASELINE MODELS AND FIVE PRUNED MODELS ON SSDD DATASET

| Model        | Recall | Precision | F1    | mAP   | FPS   | FLOPs (G) | Model Size (MB) | Parameter size (M) |
|--------------|--------|-----------|-------|-------|-------|-----------|-----------------|--------------------|
| YOLOv4       | 0.933  | 0.945     | 0.939 | 0.943 | 3.60  | 59.563    | 256.0           | 63.938             |
| YOLOv4-tiny  | 0.781  | 0.933     | 0.850 | 0.838 | 23.20 | 6.787     | 23.5            | 5.918              |
| YOLOv4-60-6  | 0.889  | 0.955     | 0.921 | 0.932 | 7.25  | 16.738    | 42.1            | 11.431             |
| YOLOv4-70-8  | 0.877  | 0.963     | 0.918 | 0.926 | 8.29  | 12.985    | 25.2            | 6.270              |
| YOLOv4-80-10 | 0.866  | 0.943     | 0.903 | 0.910 | 9.28  | 9.148     | 11.5            | 2.867              |
| YOLOv4-90-12 | 0.843  | 0.914     | 0.877 | 0.903 | 10.54 | 5.183     | 3.5             | 0.857              |
| YOLOv4-95-14 | 0.802  | 0.753     | 0.777 | 0.791 | 11.60 | 2.868     | 1.2             | 0.290              |

TABLE III  
COMPARISON OF EVALUATION METRICS UNDER FINE-TUNING AND KD ON SSDD DATASET

| Model                                | Recall | Precision | F1    | mAP   |
|--------------------------------------|--------|-----------|-------|-------|
| YOLOv4                               | 0.933  | 0.945     | 0.939 | 0.943 |
| YOLOv4-90-12<br>(Without retraining) | 0.735  | 0.828     | 0.779 | 0.779 |
| YOLOv4-90-12<br>(Fine-tuning)        | 0.843  | 0.914     | 0.877 | 0.903 |
| YOLOv4-90-12-KD                      | 0.898  | 0.907     | 0.902 | 0.931 |

It is worth noting that although the parameter size, model size, and FLOPs of YOLOv4-90-12 and YOLOv4-95-14 are much better than YOLOv4-tiny, their detection speed is significantly lower than YOLOv4-tiny. This may be due to the pruned models having a deeper network. In the detection process, the top layers of the deep network always wait for the outputs from the bottom layers to perform forward calculations, which will take more time. In the following experiments, we choose YOLOv4-90-12 model to balance the speed and accuracy of the pruned model.

3) *Effect of KD*: After pruning, the width and depth of the model are significantly reduced, but the detection accuracy will also decrease. The next step is to retrain the pruned model to improve its detection accuracy. Here we compare two retraining methods: fine-tuning and KD. In addition, the value of  $\delta$  is set to 0.5 in the process of KD, and it is obtained by multiple experiments. Retraining will not affect the model size, parameter size, FLOPs, and inference speed of the pruned models. Therefore, only four evaluation metrics of recall, precision, F1, and mAP were compared in this experiment. The comparison of evaluation metrics is given in Table III.

It can be seen that after retraining, the recall, precision, F1, and mAP of the pruned model YOLOv4-90-12 have been significantly improved. The improvements brought by fine-tuning to the above four evaluation metrics are 0.108, 0.086, 0.098, and 0.124, and they have increased by 14.69%, 10.39%, 12.58%, and 15.92%. While the improvements brought by KD are 0.163, 0.079, 0.123, and 0.152, and they have increased by 22.18%, 9.54%, 15.79%, and 19.51%. Obviously, the effect of KD is better than fine-tuning.

What is more, compared with YOLOv4, the recall, precision, F1, and mAP of YOLOv4-90-12-KD are reduced by 0.035, 0.038, 0.037, and 0.012, respectively, but the detection speed has increased to 2.93 $\times$ . We think that the significant increase

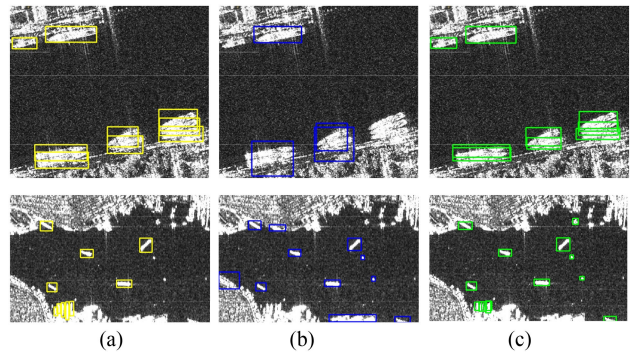


Fig. 11. Comparison results of finetune and KD. (a) Ground truth. (b) Results of finetune. (c) Results of KD.

in detection speed is worth the slight decrease in detection accuracy.

In order to compare the effects of KD and fine-tuning more intuitively, we selected two images of the nearshore area for testing, and the detection results are shown in Fig. 11. It can be seen that the missed detection of the model trained by fine-tuning is more serious. In detail, first, for the dense ship targets, the model trained by fine-tuning often recognizes the multiple ship targets as one ship (see the first row of Fig. 11). Second, the model trained by fine-tuning cannot detect the docking small-scale ship targets (see the second row of Fig. 11). However, the detection results of the model trained by KD are much better. In addition, due to the complex background of the nearshore area, both fine-tuning and KD training models will produce several false alarms.

4) *Effect of Quantization*: In fact, both FP16 and Int8 model inference need the support of hardware, and TX2 only supports FP16 model inference. Therefore, the model YOLOv4-90-12-KD is quantized from FP32 to FP16 in this experiment. In addition, we accelerated the model by TensorRT during quantization. We named the quantized model Light-YOLOv4.

Table IV lists the comparison of detection results between the quantized model and other YOLOv4 models. It can be seen that the detection speed of Light-YOLOv4 is 15.12 FPS, and the mAP is 0.930. Compared with YOLOv4, the detection speed has increased to 4.2 $\times$ , while mAP is only decreased by 0.013. After the model is quantized as FP16, the mAP is only reduced by 0.001 (compared with YOLOv4-90-12-KD), which almost has no impact on the detection results, but the detection



TABLE IV  
COMPARISON OF DETECTION PERFORMANCE AFTER MODEL QUANTIZATION  
AND ACCELERATION ON SSDD DATASET

| Model               | Data Type   | mAP          | FPS          | Acceleration Effect |
|---------------------|-------------|--------------|--------------|---------------------|
| YOLOv4              | FP32        | 0.943        | 3.60         | 1*                  |
| YOLOv4-tiny         | FP32        | 0.838        | 23.20        | 6.44*               |
| YOLOv4-90-12-KD     | FP32        | 0.931        | 10.54        | 2.93*               |
| <b>Light-YOLOv4</b> | <b>FP16</b> | <b>0.930</b> | <b>15.12</b> | <b>4.2*</b>         |

speed is increased by 4.58 FPS. It has been proved that the improvement in detection speed brought by quantization and structure optimization is very obvious.

In addition, compared with YOLOv4-tiny, the detection speed of Light-YOLOv4 is 8.08 FPS slower, but its detection accuracy is much better (the mAP is 0.092 higher).

It is worth noting that the network structure is also optimized while the model is quantized, and they are completed simultaneously. After that, the network structure and parameters of the model are fixed in the trt file. The parameter size and FLOPs of the trt file are hard to count, so we ignore the influence of quantization and structure optimization on parameter size, FLOPs, and model size.

### C. Comparison With State-of-the-Art Methods

To comprehensively evaluate Light-YOLOv4 in terms of mAP, F1, FPS, FLOPs, model size, and parameter size, we make a comparison with other target detection methods based on deep learning, including YOLO-series, SSD, Faster RCNN, FPN, and RetinaNet. The experiments are conducted on the SSDD dataset. The input image size of the YOLO series model is  $416 \times 416$ , and the input image size of other models is  $512 \times 512$ . The results are given in Table V. What is more, as shown in Fig. 12, we visualized the results in Table V so as to compare the performance of these methods more intuitively.

Obviously, YOLOv4 has the best detection accuracy, while YOLOv4-tiny has the fastest detection speed. The detection performance of other methods is between YOLOv4 and YOLOv4-tiny.

In detail, compared with YOLOv4, the detection accuracy of Light-YOLOv4 has slightly reduced, for example, its mAP has reduced by 0.013. But Light-YOLOv4's detection speed has increased by  $3.2\times$ , and FLOPs, model size, and parameter size have reduced by 91.30%, 98.63%, and 98.66%, respectively. It means that Light-YOLOv4 sacrifices a small amount of detection accuracy but saves a lot of computing resources. The detection speed of Light-YOLOv4 is slower than YOLOv4-tiny, but the remaining metrics are significantly better than YOLOv4-tiny. The above analysis means that Light-YOLOv4 has made a good balance between detection speed and accuracy.

As for the comparison with Faster RCNN, FPN, SSD, and RetinaNet, Light-YOLOv4 has the highest detection accuracy and fastest detection speed. Light-YOLOv4 is also better than the other four methods in terms of FLOPs, model size, and parameter size. It is obvious that Light-YOLOv4 is more suitable to be deployed on edge devices.

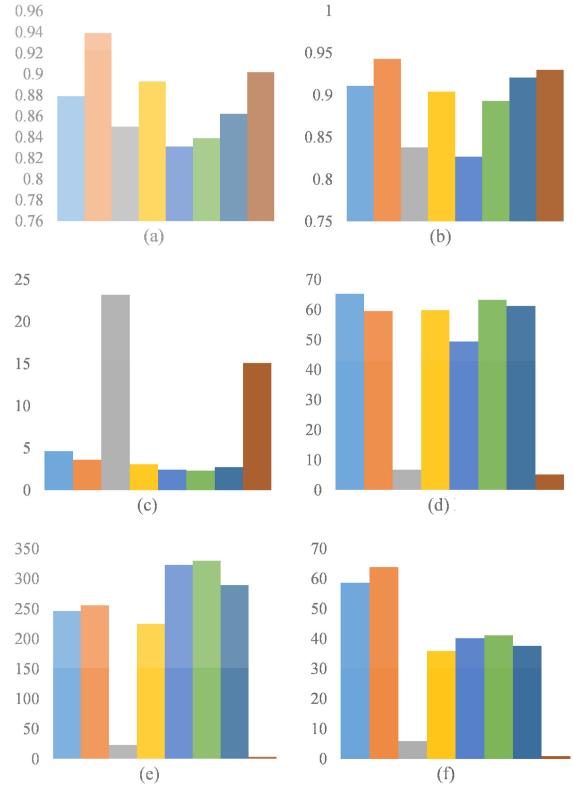


Fig. 12. Comparison of detection results under different target detection methods on the SSDD dataset. (a) F1. (b) mAP. (c) FPS. (d) FLOPs (G). (e) Model size (MB). (f) Parameter size (M).

In order to compare the detection results of these methods more intuitively, we visualized some detection results. Taking into account the limited computing resources of the edge device, we only compare YOLOv4-tiny, SSD, and Light-YOLOv4.

Fig. 13 displays the results, in which the first column shows the ground truth, while the second, third, and fourth columns show the detection results of YOLOv4-tiny, SSD, and Light-YOLOv4. What is more, the above two rows display the detection results of offshore areas, and the remaining three rows show the results of nearshore areas. It can be seen that, in the offshore areas, the detection results of the three methods are very good, and there are no false alarms and missed detections. However, in the nearshore areas with complicated backgrounds, the detection results of YOLOv4-tiny and SSD are not very good. The details are as follows.

- 1) Some small-scale ships could not be detected by YOLOv4-tiny.
- 2) For multiple dense docking ships, both YOLOv4-tiny and SSD will detect the multiple ships as one ship.
- 3) SSD will produce an obvious false alarm.

On the contrary, the detection results of Light-YOLOv4 in nearshore areas are much better; not only the dense docking ships but also the small-scale ships can be detected by our proposed model.

To summarize, compared with other target detection methods, our proposed model could meet the requirements of detection

TABLE V  
COMPARISON OF DETECTION RESULTS UNDER DIFFERENT TARGET DETECTION METHODS ON SSDD DATASET

| Model               | F1           | mAP          | FPS          | FLOPs (G)    | Model Size (MB) | Parameter size (M) | Input Size |
|---------------------|--------------|--------------|--------------|--------------|-----------------|--------------------|------------|
| YOLOv3              | 0.879        | 0.911        | 4.64         | 65.304       | 246.4           | 58.7               | 416×416    |
| YOLOv4              | 0.939        | 0.943        | 3.60         | 59.563       | 256.0           | 63.938             | 416×416    |
| YOLOv4-tiny         | 0.850        | 0.838        | 23.20        | 6.787        | 23.5            | 5.918              | 416×416    |
| SSD                 | 0.893        | 0.904        | 3.06         | 59.86        | 225.1           | 36.0               | 512×512    |
| Faster RCNN         | 0.831        | 0.827        | 1.65         | 49.37        | 323.0           | 40.2               | 512×512    |
| FPN                 | 0.839        | 0.893        | 1.43         | 63.25        | 330.2           | 41.1               | 512×512    |
| RetinaNet           | 0.862        | 0.921        | 2.13         | 61.22        | 290.0           | 37.7               | 512×512    |
| <b>Light-YOLOv4</b> | <b>0.902</b> | <b>0.930</b> | <b>15.12</b> | <b>5.183</b> | <b>3.5</b>      | <b>0.857</b>       | 416×416    |

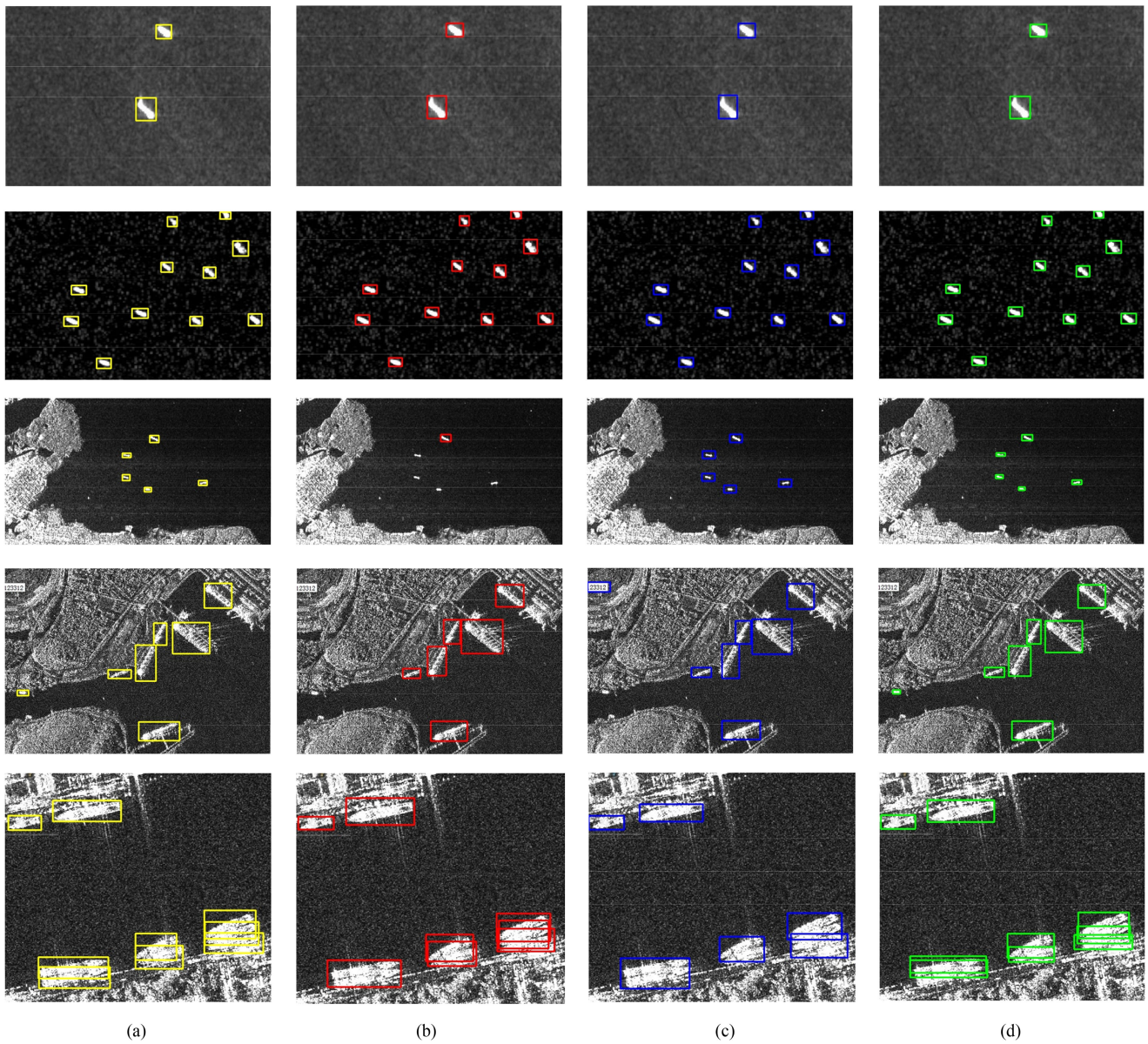


Fig. 13. Comparison detection results of YOLOv4-tiny, SSD, and Light-YOLOv4. (a) Ground truth. (b) Detection results of YOLOv4-tiny. (c) Detection results of SSD. (d) Detection results of Light-YOLOv4.



TABLE VI  
COMPARISON OF DETECTION RESULTS ON GAOFEN AIRPLANE DATASET

| Model        | Recall | Precision | F1    | mAP   |
|--------------|--------|-----------|-------|-------|
| YOLOv4       | 0.907  | 0.749     | 0.821 | 0.891 |
| Light-YOLOv4 | 0.911  | 0.728     | 0.809 | 0.885 |

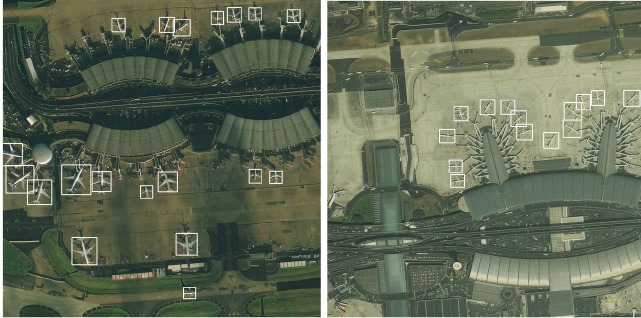


Fig. 14. Detection results of Light-YOLOv4 on Gaofen Airplane dataset.

accuracy and detection speed at the same time and it is more suitable for edge devices with limited computing resources.

#### D. Detection Results on Gaofen Airplane Dataset

In order to verify the feasibility of our proposed method in optical remote sensing target detection, we carried out this experiment. The image size in the Gaofen Airplane dataset is large, so we crop the images ( $1024 \times 1024$ ) into several small images ( $416 \times 416$ ) with an overlap of 100 pixels. After the target detection is completed, several small images will be spliced into a large image. This image processing method is widely used in the field of remote sensing.

Here, we only compare Light-YOLOv4 with YOLOv4 (the Light-YOLOv4 in this experiment is also obtained based on YOLOv4-90-12). What is more, only four metrics were compared, and they are recall, precision, F1, and mAP. The detection results are given in Table VI.

It can be found that compared to YOLOv4, the mAP and F1 of Light-YOLOv4 have decreased by 0.006 and 0.012, respectively, which has little impact on the detection accuracy. At the same time, the detection speed is  $4.2 \times$  that of YOLOv4. What is more, we have visualized some detection results of Light-YOLOv4, which are shown in Fig. 14. Obviously, Light-YOLOv4 could detect these targets well, even if the airplane is small or the background is complex. The above experiments show that our proposed method still has good performance in optical remote sensing target detection.

#### V. CONCLUSION

Target detection methods based on deep learning have high computational complexity and memory consumption, which makes them difficult to deploy on edge devices with limited resources. To solve this problem, this article proposed to learn a lightweight detector through model compression. We select YOLOv4 as the baseline and then continue to implement sparsity training, channel and layer pruning, KD, and quantization on it to

obtain the lightweight detector, that is, Light-YOLOv4. Finally, Light-YOLOv4 was successfully deployed on NVIDIA Jetson TX2, which is an edge device with low power consumption and high performance.

To verify the effectiveness of Light-YOLOv4, several experiments were carried out on the SSDD dataset. The results show that Light-YOLOv4 is able to achieve comparable detection accuracy as YOLOv4. At the same time, the parameter size is reduced by 98.66%, the model size is reduced by 98.63%, FLOPs is reduced by 91.30%, and the inference speed is increased to  $4.2 \times$ . Comparison experiments with other state-of-the-art methods show that our proposed model can meet the requirements of detection accuracy and speed at the same time. In addition, experiments on the Gaofen Airplane dataset verify the feasibility of our method in optical remote sensing target detection. Of course, there are still some limitations to our method. For example, the training process of our model is time-consuming, and the pruned model cannot make full use of the computing resource. In future work, we will try other edge devices and other model compression and acceleration methods to further improve the detection performance.

#### REFERENCES

- [1] X. Sun *et al.*, "Progress and challenges of remote sensing edge intelligence technology," *J. Image Graph.*, vol. 25, no. 9, pp. 1719–1738, 2020.
- [2] M. Amami, A. Ghorbanian, S. A. Ahmadi, M. Kakooei, A. Moghimi, and S. M. Mirmazloumi, "Google Earth Engine cloud computing platform for remote sensing big data applications: A comprehensive review," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 5326–5350, Sep. 2020.
- [3] Y. Yao, Z. Jiang, H. Zhang, and Y. Zhou, "On-board ship detection in micro-nano satellite based on deep learning and COTS component," *Remote Sens.*, vol. 11, no. 7, Apr. 2019, Art. no. 762.
- [4] S. Minaeian, J. Liu, and Y.-J. Son, "Effective and efficient detection of moving targets from a UAV's camera," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 2, pp. 497–506, Jan. 2018.
- [5] A. Bouguettaya, H. Zarzour, A. Kechida, and A. M. Taberkit, "Vehicle detection from UAV imagery with deep learning: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2021.3080276.
- [6] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019.
- [7] H. Wang, Y. Yu, Y. Cai, X. Chen, L. Chen, and Q. Liu, "A comparative study of state-of-the-art deep learning algorithms for vehicle detection," *IEEE Intell. Transp. Syst. Mag.*, vol. 11, no. 2, pp. 82–95, Mar. 2019.
- [8] M. Zalpour, G. Akbarizadeh, and N. Alaei-Sheini, "A new approach for oil tank detection using deep learning features with control false alarm rate in high-resolution satellite imagery," *Int. J. Remote Sens.*, vol. 41, pp. 2239–2262, Mar. 2019.
- [9] Z. Gao, H. Ji, T. Mei, B. Ramesh, and X. Liu, "EOVNet: Earth-observation image-based vehicle detection network," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 9, pp. 3552–3561, Aug. 2019.
- [10] Z. Cui, X. Wang, N. Liu, Z. Cao, and J. Yang, "Ship detection in large-scale SAR images via spatial shuffle-group enhance attention," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 1, pp. 379–391, Jun. 2021.
- [11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.
- [12] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6517–6525.
- [13] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1–6.
- [14] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal speed and a accuracy of object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 198–215.

- [15] W. Liu, D. Anguelov, and D. Erhan, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.
- [16] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2999–3007.
- [17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
- [18] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 936–944.
- [19] L. Ma, Y. Liu, X. Zhang, Y. Ye, G. Yin, and B. A. Johnson, "Deep learning in remote sensing applications: A meta-analysis and review," *ISPRS J. Photogrammetry Remote Sens.*, vol. 152, pp. 166–177, Jul. 2019.
- [20] X. Zhu et al., "Deep learning meets SAR: Concepts, models, pitfalls, and perspectives," *IEEE Geosci. Remote Sens. Mag.*, to be published, doi: [10.1109/MGRS.2020.3046356](https://doi.org/10.1109/MGRS.2020.3046356).
- [21] R. Dong, D. Xu, J. Zhao, L. Jiao, and J. An, "Sig-NMS-based faster R-CNN combining transfer learning for small target detection in VHR optical remote sensing imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 11, pp. 8534–8545, Nov. 2019.
- [22] Z. Cui, Q. Li, Z. Cao, and N. Liu, "Dense attention pyramid networks for multi-scale ship detection in SAR images," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 11, pp. 8983–8997, Nov. 2019.
- [23] Q. An, Z. Pan, L. Liu, and H. You, "DRBox-v2: An improved detector with rotatable boxes for target detection in SAR images," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 11, pp. 8333–8349, Nov. 2019.
- [24] Q. Li, L. Mou, Q. Xu, Y. Zhang, and X. X. Zhu, "R3-Net: A deep network for multioriented vehicle detection in aerial images and videos," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 7, pp. 5028–5042, Jul. 2019.
- [25] X. Wang, Z. Cui, Z. Cao, and S. Dang, "Dense docked ship detection via spatial group-wise enhance attention in SAR images," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2020, pp. 1244–1247.
- [26] H. Gao, Y. L. Tian, F. Y. Xu, and S. Zhong, "Survey of deep learning model compression and acceleration," *J. Softw.*, vol. 32, no. 1, pp. 68–92, 2021.
- [27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Int. Conf. Neural Inf. Proc. Syst.*, 2012, pp. 1097–1105.
- [28] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1800–1807.
- [29] Y. Li, S. Zhang, and W.-Q. Wang, "A lightweight faster R-CNN for ship detection in SAR images," *IEEE Geosci. Remote Sens. Lett.*, to be published, doi: [10.1109/LGRS.2020.3038901](https://doi.org/10.1109/LGRS.2020.3038901).
- [30] Y. Zhao, Y. Yin, and G. Gui, "Lightweight deep learning based intelligent edge surveillance techniques," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1146–1154, Dec. 2020.
- [31] P. Ding, Y. Zhang, W.-J. Deng, P. Jia, and A. Kuijper, "A light and faster regional convolutional neural network for object detection in optical remote sensing images," *ISPRS J. Photogrammetry Remote Sens.*, vol. 141, pp. 208–218, Jul. 2018.
- [32] M. Zhou, Z. Zou, Z. Shi, W.-J. Zeng, and J. Gui, "Local attention networks for occluded airplane detection in remote sensing images," *IEEE Geosci. Remote Sens. Lett.*, vol. 17, no. 3, pp. 381–385, Mar. 2020.
- [33] D. Xiao, F. Shan, Z. Li, B. T. Le, X. Liu, and X. Li, "A target detection model based on improved tiny-Yolov3 under the environment of mining truck," *IEEE Access*, vol. 7, pp. 123757–123764, 2019.
- [34] B. L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proc. IEEE*, vol. 108, no. 4, pp. 485–532, Apr. 2020.
- [35] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2755–2763.
- [36] J. Guo, W. Zhang, W. Ouyang, and D. Xu, "Model compression using progressive channel pruning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 3, pp. 1114–1124, Mar. 2021.
- [37] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*. [Online]. Available: <https://arxiv.org/abs/1503.02531>
- [38] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," 2014, *arXiv:1412.6550*. [Online]. Available: <https://arxiv.org/abs/1412.6550>
- [39] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4820–4828.
- [40] Y. Xu, W. Dai, Y. Qi, J. Zou, and H. Xiong, "Iterative deep neural network quantization with lipschitz constraint," *IEEE Trans. Multimedia*, vol. 22, no. 7, pp. 1874–1888, Jul. 2020.
- [41] TensorRT, NVIDIA, "NVIDIA Deep learning SDK[DB/OL]," Nov. 2019. [Online]. Available: <https://docs.nvidia.com/deeplearning/sdk/index.html>
- [42] J. Li, C. Qu, S. Peng, and B. Deng, "Ship detection in SAR images based on convolutional neural network," *Syst. Eng. Electron.*, vol. 40, no. 9, pp. 1953–1959, 2018.
- [43] J. M. P. Nascimento, M. P. Vestias, and G. Martin, "Hyperspectral compressive sensing with a system-on-chip FPGA," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 3701–3710, May 2020.
- [44] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I. H. Yeh, "CSPNet: A new backbone that can enhance learning capability of CNN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 1571–1580.
- [45] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8759–8768.
- [46] W. Chen, H. Wang, H. Li, Q. Li, Y. Yang, and K. Yang, "Real-time garbage object detection with data augmentation and feature fusion using UAV low-altitude remote sensing images," *IEEE Geosci. Remote Sens. Lett.*, to be published, doi: [10.1109/LGRS.2021.3074415](https://doi.org/10.1109/LGRS.2021.3074415).
- [47] S. Du, P. Zhang, B. Zhang, and H. Xu, "Weak and occluded vehicle detection in complex infrared environment based on improved YOLOv4," *IEEE Access*, vol. 9, pp. 25671–25680, 2021.
- [48] P. Zhang, Y. Zhong, and X. Li, "SlimYOLOv3: Narrower, faster, and better for real-time UAV applications," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop*, 2019, pp. 37–45.
- [49] E. M. Khayrov, M. Y. Malsagov, and I. M. Karandashev, "Post-training quantization of deep neural network weights," in *Proc. 21st Int. Conf. Neuroinform.*, 2019, pp. 230–238.
- [50] Y. Zhao, L. Zhao, B. Xiong, and G. Kuang, "Attention receptive pyramid network for ship detection in SAR images," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 2738–2756, May 2020.
- [51] X. Wang, Z. Cui, Z. Cao, and S. Dang, "Dense docked ship detection via spatial group-wise enhance attention in SAR images," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2020, pp. 1244–1247.
- [52] G. Verma, Y. Gupta, A. M. Malik, and B. Chapman, "Performance evaluation of deep learning compilers for edge inference," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops*, 2021, pp. 858–865.
- [53] E. Jeong, J. Kim, S. Tan, J. Lee, and S. Ha, "Deep learning inference parallelization on heterogeneous processors with TensorRT," *IEEE Embedded Syst. Lett.*, to be published, doi: [10.1109/LES.2021.3087707](https://doi.org/10.1109/LES.2021.3087707).
- [54] F. Zhang, Y. Liu, Y. Zhou, Q. Yin, and H.-C. Li, "A lossless lightweight CNN design for SAR target recognition," *Remote Sens. Lett.*, vol. 11, pp. 485–494, May 2020.
- [55] H. Chen, F. Zhang, B. Tang, Q. Yin, and X. Sun, "Slim and efficient neural network design for resource-constrained SAR target recognition," *Remote Sens.*, vol. 10, Oct. 2018, Art. no. 1618, doi: [10.3390/rs10101618](https://doi.org/10.3390/rs10101618).
- [56] 2017. [Online]. Available: <http://on-demand.gptechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf>
- [57] "2020 Gaofen challenge on automated high-resolution Earth observation image interpretation," 2020. [Online]. Available: <http://en.sw.chreos.org>
- [58] Z. Sun et al., "An anchor-free detection method for ship targets in high-resolution SAR images," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 7799–7816, Aug. 2021.



**Xiaojie Ma** received the B.S. in measurement and control technology and instrument, the M.S. degree in measuring and testing technologies and instruments from Yanshan University, Qinhuangdao, China, in 2016 and 2019, respectively. He is currently working toward the Ph.D. degree with the State Key Laboratory of Complex Electromagnetic Environment Effects, National University of Defense Technology, Changsha, China.

His current research interests include object detection, SAR ATR, and machine learning.





**Kefeng Ji** (Member, IEEE) received the B.S. degree in aerospace engineering from Northwestern Polytechnical University, Xi'an, China, in 1996, and the M.S. and Ph.D. degrees in information and telecommunication engineering from the National University of Defense Technology (NUDT), Changsha, China, in 1999 and 2003, respectively.

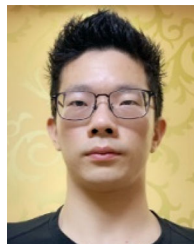
In 2003, he joined the College of Electronic Science and Technology, NUDT, where he is currently a Professor. He has authored or coauthored more than 80 papers. His current research interests include signal processing, machine learning, pattern recognition, remote sensing information processing, SAR image interpretation, target detection, recognition, feature extraction, and marine surveillance.

Dr. Ji is a reviewer of several international journals and conferences, such as *IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING*, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *IEEE Geoscience and Remote Sensing Letters*, *IEEE Access*, *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS*, *International Journal of Remote Sensing*, *Remote Sensing Letters*, *European Journal of Remote Sensing*, and *Remote Sensing*.



**Boli Xiong** received the B.S. degree in electronic engineering, the M.S. degree in photogrammetry and remote sensing, and the Ph.D. degree in communication engineering from the National University of Defense Technology, Changsha, China, in 2004, 2006, and 2012, respectively.

He is currently an Associate Professor with the School of Electronic Science, National University of Defense Technology. His current research interests include SAR/PolSAR image registration and change detection, SAR automatic target recognition, pattern recognition, and machine learning.



**Linbin Zhang** received the B.S. degree in information engineering in 2018 from the National University of Defense Technology, Changsha, China, where he is currently working toward the Ph.D. degree with the State Key Laboratory of Complex Electromagnetic Environment Effects.

His current research interests include object detection, image classification, few-shot learning, and machine learning and its applications to remote sensing images.



**Sijia Feng** received the B.S. and M.S. degrees in information engineering in 2016 and 2019, respectively, from the National University of Defense Technology, Changsha, China, where she is currently working toward the Ph.D. degree with the State Key Laboratory of Complex Electromagnetic Environment Effects.

Her current research interests include SAR image interpretation, feature extraction, and machine learning.



**Gangyao Kuang** (Senior Member, IEEE) received the B.S. and M.S. degrees in geophysics from the Central South University of Technology, Changsha, China, in 1988 and 1991, and the Ph.D. degree in communication and information from the National University of Defense Technology, Changsha, China, in 1995.

He is currently a Professor with the School of Electronic Science, National University of Defense Technology. His research interests include remote sensing, SAR image processing, change detection,

SAR ground moving target indication, and classification with polarimetric SAR images.