remote sensing

MDPI

Article

Hardware-Aware Design of Speed-Up Algorithms for Synthetic Aperture Radar Ship Target Detection Networks

Yue Zhang [1,2], Shuai Jiang [1], Yue Cao [1,2], Jiarong Xiao [1], Chengkun Li [1,2], Xuan Zhou [1] and Zhongjun Yu [1,2,*]

[1] Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100190, China; zhangyue20@mails.ucas.ac.cn (Y.Z.); jiangshuai@aircas.ac.cn (S.J.); yuecao0927@hotmail.com (Y.C.); xiaojiarong@aircas.ac.cn (J.X.); lichengkun20@mails.ucas.ac.cn (C.L.); zhouxuan@aircas.ac.cn (X.Z.)

[2] School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 101408, China

* Correspondence: yuzj@ucas.ac.cn

Abstract: Recently, synthetic aperture radar (SAR) target detection algorithms based on Convolutional Neural Networks (CNN) have received increasing attention. However, the large amount of computation required burdens the real-time detection of SAR ship targets on resource-limited and power-constrained satellite-based platforms. In this paper, we propose a hardware-aware model speed-up method for single-stage SAR ship targets detection tasks, oriented towards the most widely used hardware for neural network computing—Graphic Processing Unit (GPU). We first analyze the process by which the task of detection is executed on GPUs and propose two strategies according to this process. Firstly, in order to speed up the execution of the model on a GPU, we propose SAR-aware model quantification to allow the original model to be stored and computed in a low-precision format. Next, to ensure the loss of accuracy is negligible after the acceleration and compression process, precision-aware scheduling is used to filter out layers that are not suitable for quantification and store and execute them in a high-precision mode. Trained on the dataset HRSID, the effectiveness of this model speed-up algorithm was demonstrated by compressing four different sizes of models (yolov5n, yolov5s, yolov5m, yolov5l). The experimental results show that the detection speeds of yolov5n, yolov5s, yolov5m, and yolov5l can reach 234.7785 fps, 212.8341 fps, 165.6523 fps, and 139.8758 fps on the NVIDIA AGX Xavier development board with negligible loss of accuracy, which is 1.230 times, 1.469 times, 1.955 times, and 2.448 times faster than the original before the use of this method, respectively.

Keywords: synthetic aperture radar (SAR); target detection; convolutional neural networks (CNN); model speed-up algorithms; graphic processing unit (GPU)

Citation: Zhang, Y.; Jiang, S.; Cao, Y.; Xiao, J.; Li, C.; Zhou, X.; Yu, Z. Hardware-Aware Design of Speed-Up Algorithms for Synthetic Aperture Radar Ship Target Detection Networks. *Remote Sens.* **2023**, *15*, 4995. https://doi.org/10.3390/rs15204995

Academic Editors: Guangcai Sun, Jiang Qian, Lei Yang and Jinsong Zhang

Received: 13 September 2023
Revised: 11 October 2023
Accepted: 13 October 2023
Published: 17 October 2023

1. Introduction

Synthetic aperture radar (SAR) has an irreplaceable and unique advantage in the field of remote sensing because of its all-day and all-weather ability to provide reliable remote sensing information. Active detection [1] and identification [2] of targets of interest has been an important issue for synthetic aperture radar (SAR) systems. And among them, SAR ship target detection, as a very critical step in SAR data processing, is of great importance in both military and civilian fields.

Traditional detection algorithms such as threshold methods [3], statistical methods [4], and transformation methods [5] rely on information such as predefined image features or distribution characteristics and are difficult to apply in different scenarios. As the most widely used traditional detection algorithm, the constant false alarm rate (CFAR) method [6,7] accomplishes the ship target detection task by comparing the statistical features of a region with a set threshold. However, in complex backgrounds, it is difficult to characterize the scattering properties of ship targets and clutter using simple statistical methods, which results in the degradation of detection performance. In addition, the

computation of distribution parameter estimation in CFAR takes a significant amount of time, which makes real-time detection very difficult.

The rapidly developing deep learning techniques provide a new way for ship target detection in SAR images and have been applied by many researchers. Benefiting from the good robustness of neural networks and their ability to handle complex tasks, ship target detection algorithms using neural networks have achieved good results. However, SAR ship target detection tasks often lead to problems such as missed and false detection of targets due to complex backgrounds, poor image quality, and inconsistent target scales. In order to overcome these problems and improve the accuracy of SAR image ship target detection, researchers have made improvements on both two-stage [8–15] and single-stage [16–21] networks, respectively. Compared to single-stage detection networks, two-stage-based SAR ship target detection networks have been studied earlier. The two-stage network first extracts the object region and then recognizes the region by CNN classification. Nie et al. [9] improved the FPN structure based on Mask-RCNN and added an attention mechanism to improve the detection accuracy. To improve the detection accuracy of multi-scale ship targets, Zhao et al. [10] proposed a two-stage attention receptive pyramid network (ARPN). Cui et al. [11] proposed Dense Attention Pyramid Networks using a pyramid structure and an attention mechanism approach to improve the performance of the network in detecting small-scale targets. The Balanced Feature Pyramid Network proposed by Zhang et al. [12] was used to solve the problem of low detection rate due to the uneven distribution of ships in SAR images. Lv et al. [13] used the information from single-look SAR images to reconstruct the features of the ship target based on the original detector. Ai et al. [14] combined CFAR, Convolutional Neural Network, and Support Vector Machine (SVM) to achieve the task of multi-target detection in complex backgrounds. The single-stage SAR ship target detection networks do not need to reach the suggestion box stage, which directly generates the class probability and position coordinate values of the object. This type of network integrates regression of detection frames and classification of objects. Yang Xi et al. [16] proposed the Receptive Field Increase Module (RFIM) to solve the multi-scale problem of ship targets in SAR images. Chen et al. [18] used an end-to-end detection network and a number of augmentation strategies and conducted experiments on both offshore and distant sea datasets. Guo et al. [20] added Convolutional Block Attention Module (CBAM) and Bidirectional Feature Pyramid Network (BIFPN) to yolov5 to improve detection performance. All these methods have greatly improved the detection accuracy in SAR ship target detection, but these network models are often complex, resulting in slower detection speed.

In order to reduce the complexity of network models and increase the speed of model detection, researchers are dedicated to improving the first stage of detection networks and have developed a variety of lightweight detection models [22–29]. Jiang et al. [23] accelerated the network detection speed by cropping the yolov4-lite network while improving the detection accuracy by considering the characteristics of three-channel images. Zhou et al. [24] used small convolutions and a customized extractor network to reduce the computational and parametric size of the network model. Ren et al. [29] designed a Lightweight Network, which uses a lightweight feature enhancement backbone to reduce the amount of computation and a channel and position enhancement attention module to improve detection accuracy. Zhang et al. [24] proposed a hybrid representation learning-enhanced SAR target detection algorithm by taking into account the unique features of SAR images and the lightweight structure of the network. Although these studies have targeted the parameters and computational volume of the algorithms for optimization and are designed to improve the detection speed by designing network structures, there are still some issues that have not been fully studied. In practical applications, SAR systems need to be simultaneously highly mobile, highly contingent, and highly accurate. In order to achieve real-time target detection algorithms, many scholars have proposed running this part of the work on airborne or satellite-based edge hardware platforms, which have constrained computing resources and power consumption. Therefore, reasonable arrangement of the

limited computational resources on the processing platform in order to further improve the detection speed of SAR ship target detection networks has become the goal of researchers.

Recently, some hardware manufacturers, such as NVIDIA, AMD, INTEL, etc., have designed their own model speed-up methods to achieve deep acceleration from their own hardware. Li et al. [30], working at Intel, proposed the Analytical Clipping for Integer Quantization (ACIQ) method and gave an analytical solution for the optimal truncation value on this basis. Markus Nagel et al. [31] introduced two algorithms, Post-training quantization (PTQ) and Quantization-aware-training (QAT), to minimize the loss of accuracy caused by model compression. Wu et al. [32] proposed quantization techniques suitable for high-throughput hardware accelerators. Based on these approaches, NVIDIA proposes its own applicable GPU acceleration method for model inference tasks. Unfortunately, when using this model speed-up method in some SAR ship target detection tasks running on platforms with limited computing resources and power consumption, there is still a huge demand for improving accuracy and speed. First of all, the operation speed of the detection network is improved with the use of the method for the network, but at the cost of a certain loss of accuracy. It is widely acknowledged that accuracy is more important than speed as the most important metric in SAR ship target detection tasks. Therefore, this method of increasing speed at the expense of accuracy is not desirable. Secondly, this method works well for speed improvement in small and medium sized networks, but the speed increase after using this method for larger network structures is negligible, which limits its application to some extent. In summary, how to accelerate SAR ship target detection networks deployed on edge platforms with limited computational resources and power consumption, without much loss of detection accuracy, is still a great challenge.

To address this challenge, a novel model speed-up method applicable to SAR ship target detection algorithms is proposed in this paper, using embedded GPUs as the target deployment platform. The main contributions of this paper are as follows:

- We propose a GPU-oriented model speed-up method for SAR ship target detection tasks and use this algorithm for yolov5n, yolov5s, yolov5m, and yolov5l. Then, the accelerated models are deployed on an NVIDIA AGX Xavier development board.
- This paper provides the first detailed analysis of the execution of the inference process for the SAR ship target detection task on a GPU. The results of the analysis give direction and provide a basis for subsequent optimization.
- To reduce the computation time on the GPU, SAR-aware model quantification is used. Firstly, we introduced a low-precision quantization method to improve the detection speed. After quantization, we used a SAR-adapting Calibration method to reduce the accuracy loss due to quantization while performing computational acceleration. At the same time, we used activation layer merging to reduce the number of memory readings to speed up the network.
- To further reduce the accuracy loss, the Precision-aware scheduling method is proposed. This method minimizes accuracy loss while guaranteeing speed-up by inverse quantization of layers unsuitable for quantization.

The rest of this study is as follows: Section 2 introduces the steps of a SAR target detection algorithm inference task running on a heterogeneous system including CPU and GPU. Section 3 introduces the SAR-aware model quantification method to reduce the time of inference tasks running on heterogeneous systems. On this basis, a precision-aware scheduling method is proposed according to the GPU computing power characteristics of the heterogeneous system to reduce the precision loss caused by acceleration. In Section 4, we use the method in Section 3 on four different networks to verify the effectiveness of the proposed method. The method used in this paper is compared with TensorRT, the most commonly used GPU acceleration method, and the effectiveness of the proposed method is proved. In Section 5, the experimental results are analyzed and discussed. Finally, the main conclusions are drawn in Section 6.

2. Analysis

The inference process of the SAR ship target detection task usually runs in a heterogeneous system containing a CPU and a GPU, and its structure is shown in Figure 1. When performing computational tasks, the GPU cannot run alone and usually requires the CPU to assist in running. CPUs usually contain multiple processor cores and caches, each of which can execute one or more threads, and the main advantage of CPUs is that they can handle different types of complex tasks and can better manage and allocate system resources to ensure smooth execution of computational tasks, while GPUs have their own unique streaming multiprocessor (SM) computing architecture and higher bandwidth memory, better suited to handle large amounts of data and parallel computing tasks. The two units are connected via a peripheral component interconnect express (PCI-Express) bus to complement each other and improve the performance and efficiency of the whole system.

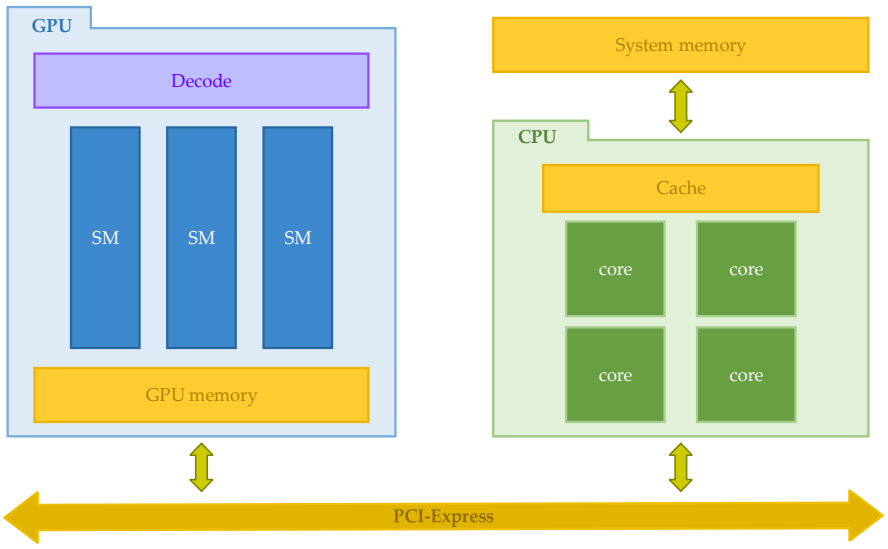


Figure 1. The heterogeneous system.

Although the computation in the inference task is performed on the GPU, the GPU first needs to obtain instructions and data from the CPU and return the results to the CPU after the computation is completed. The whole inference task is shown in Figure 2, which can be divided into four stages.

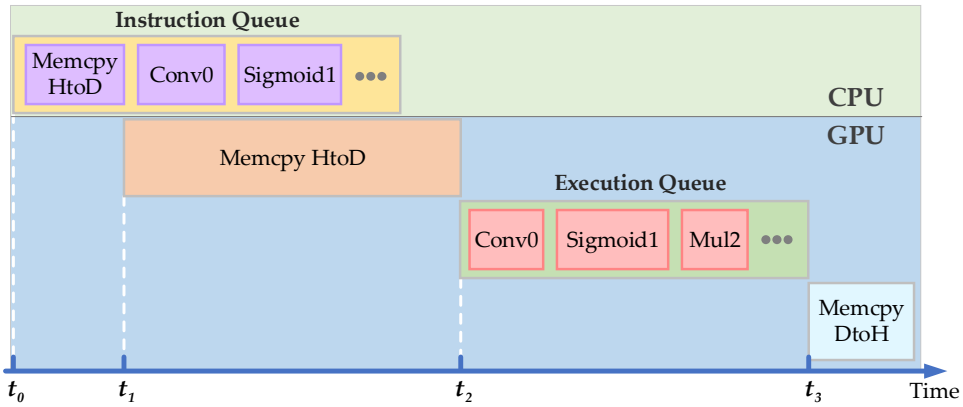


Figure 2. The inference process of the SAR ship target detection task.

- The CPU compiles the executed code into instructions that can be executed by the GPU and sends them to the GPU through the instruction queue, including data loading instructions, arithmetic computation instructions, and data result transfer instructions.
- As shown in Figure 2, when the GPU receives the instruction of data loading, it starts the work of data loading and loads the image data saved in the system memory to the memory in the GPU at the moment of t_1.
- Once the data and instructions are ready at the moment of t_2., the arithmetic computation starts. In this process, the GPU reads the data from the GPU's memory to start the operation of the convolution, activation, and other arithmetic operations. Each arithmetic operation reads input data from the memory on the GPU, and then saves the output data on the GPU's memory after the computation is completed.
- When the above operations are finished at the moment of t_3, the results will be saved in the memory of the GPU. Since the post-processing needs to be performed in the CPU, the results will be reloaded into the system memory.

3. Method

Figure 3 illustrates the frame of work in this paper, which aims to ensure real-time and accurate performance of detection algorithms running on the edge platform. Based on a detailed analysis of the detection inference process running on an edge GPU platform, we propose corresponding hardware-aware speed-up algorithms for SAR ship target detection networks.

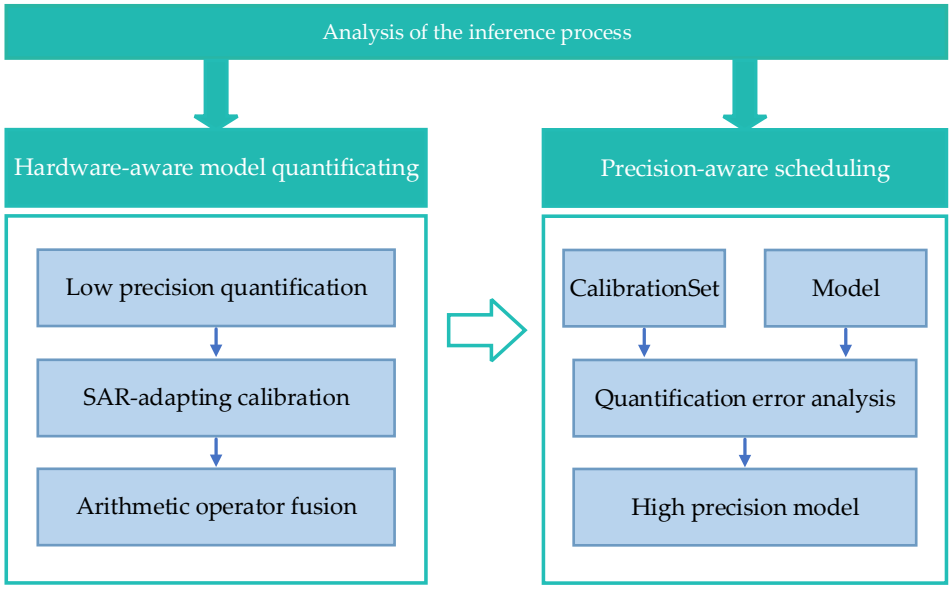


Figure 3. The frame of work in this paper.

In order to speed up the computation of the arithmetic operator, the SAR-aware model quantification method is used on the trained model. A SAR image detection model-friendly calibration algorithm is used based on low-bit quantization. Also, considering the access demand during the arithmetic operator computation, the arithmetic operator fusion method is used to reduce the number of memory accesses and effectively shorten the arithmetic operator computation time.

Considering the loss of model accuracy caused by the quantization process, this paper uses precision-aware scheduling after quantization to inverse-quantize the layers that are not suitable for quantization and continue to use the FP32 format for computation, thus compensating for the loss of accuracy.

3.1. SAR-Aware Model Quantification

In the inference analysis of target detection, the longest time occupation is often the time of arithmetic operations. Previous studies have used lightweight models to reduce the number of operations but have not optimized this process from the perspective of underlying hardware execution. Although hardware vendors have proposed some relevant model speed-up algorithms from the hardware point of view, the direct use of these model speed-up algorithms in the SAR image detection model mountain will result in accuracy degradation. In order to achieve high model acceleration with finite accuracy loss, we propose a SAR-aware model quantification, which consists of three parts: low precision quantification, SAR-adapting calibration, and arithmetic operator fusion.

3.1.1. Low Precision Quantification

In order to ensure the accuracy of the final detection in the target detection network of SAR images, the weights and feature maps are stored and calculated using high-precision floating-point numbers. Most of them are stored and calculated in the form of FP32 in the deployed hardware. However, two benefits can be brought about if they are stored and calculated in the format of INT8.

Firstly, using fewer bits for data reading, writing, and storage will effectively increase the speed of data reading and writing before and after each arithmetic computation. Secondly, the peak computational power of INT8 computation in many GPUs is much larger than that of FP32 computation, which means that the computational speed of INT8 is much higher than that of FP32. Both of these can accelerate the network inference computation. Therefore, we use the 8-bit linear layer-by-layer quantization algorithm to speed-up the network model and achieve the speed-up of network inference.

The quantization calculation of the operator is shown in Figure 4, where weight_fp32 and feature map_fp32 represent the input weights and feature map for each calculation, and s_w and s_fm represent the quantization parameters when the value conversion is performed. Before each calculation starts, it is necessary to map the high-precision value stored in FP32 to an INT8 format value in the interval $[-128, 127]$ using the following formula:

$$x = round(\frac{xfp32}{s}) \tag{1}$$

$$x_{\text{int8}} = clamp(-128, 127, x) \tag{2}$$

where s denotes the scaling factor. "*Round*" represents the rounding function, and the resulting floating-point value is rounded to an integer according to the number after the decimal point. "*Clamp*" is a truncation function, meaning that integers less than -128 will become -128, and integers greater than 127 will be evaluated as 127.

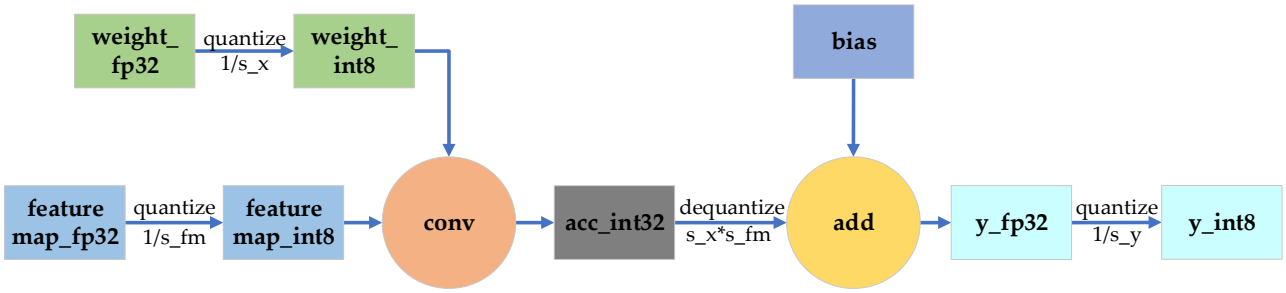


Figure 4. The quantization calculation of the operator.

After convoluting the x in INT8 format with the w in INT8 format, the convolution result is multiplied with the product of s_w and s_fm. Then, the obtained intermediate result is unquantized and added to the bias without quantization. Finally, the result is output quantized.

3.1.2. SAR-Adapting Calibration

The quantization parameters are the decisive factors that affect the accuracy of the quantized network. The formula for calculating the quantization parameters is shown below:

$$s = \frac{T_1 - T_2}{\max(\text{int}) - \min(\text{int})} \tag{3}$$

where T_1, T_2 denote the truncated range of the floating point number.

For the quantization of weights, since the weight is already a fixed value that does not change after the training of the network, and also since there are fewer values in each weight, we can directly use the maximum and minimum values in each weight as the truncated range of float point numbers.

As for the activation values (feature map of each layer), since the values generated by each inference are different, the maximum and minimum values generated by a certain inference cannot simply be used as the truncation threshold.

An analytical solution for minimizing the quantization error was calculated in article [30] using a detailed formula derivation, but it assumes a Gaussian distribution for the activation values. Nevertheless, for the activation values in the SAR ship target detection network, it is not possible to strictly prove that the activation values are Gaussian distributed using the formula. The method used by NVIDIA [32] uses kullback-leibler (KL) divergence as a measure of quantization error, which is also not adapted to the SAR ship target detection network. The reason for this is as follows: the detection task can often be subdivided into a regression task for the detection frame and a classification task for the target, and there are numerous types of targets to be detected in optical images. The two datasets we commonly use are PASCAL VOC and coco, which have 20 and 91 classes of targets, respectively. Therefore, the classification task is quite important in the detection network of optical images. However, for ship target detection tasks for SAR images, there is only one type of target for classification. Thus, for this type of detection task, more attention should be given to the regression task in detection. However, kullback-leibler (KL) divergence is concerned with the difference between the two tensor distributions of input and output, while regression is more concerned with the difference between the values of these two distributions. Consequently, the error calculation of kullback-leibler (KL) divergence is not suitable for the regression tasks.

In order to solve the above two drawbacks, this paper proposes a calibration algorithm suitable for SAR ship target detection tasks. The algorithm adopts a statistically based approach to avoid the problem of assuming the distribution of the activation values, and it uses Mean Squared Error (MSE) that pays more attention to the values before and after quantization as a measure of quantization error, making it more adaptable to the target detection task of SAR.

The specific steps of the proposed algorithm are as follows:

1. Prepare a trained SAR ship target detection model.
2. Select a subset from the SAR image validation set as the calibration set.
3. Inference is performed on the model of FP32 using the calibration dataset.
4. The inference process traverses each layer of the network.
5. The calibration set is deduced on the existing network and the histograms of activation values can be obtained which are further divided into 2048 intervals.
6. The median of each of these intervals is chosen as the T-value.
7. Iterate over different thresholds T and select the T that makes the MSE obtain the minimum value, which can be represented as

$$\text{Error}_{\text{mse}} = \frac{\sum_{i}^{N} (x_i - x_{\text{int8}})^2}{N} \tag{4}$$

where x_i denotes the value of each element of the tensor and N denotes the number of elements of the tensor.

8. Return a series of T values, one for each layer T, creating the Calibration Table.
9. The quantization parameter S is calculated for each layer based on the value of T.

Using this method, the accuracy error of the SAR ship target detection network is minimized without considering the feature map distribution.

3.1.3. Arithmetic Operator Fusion

To further reduce the time of arithmetic operator computation, the arithmetic operator fusion approach is adopted in this paper.

A convolutional neural network can be regarded as a computational graph consisting of several nodes, where an arithmetic operator is the smallest scheduling unit of the neural network scheduling. Arithmetic operator fusion is designed to fuse the original multiple operators into a new operator, and this operation can accomplish the following two aspects of optimization.

In heterogeneous systems, the execution of each operator requires the CPU to first issue a computational task. The GPU starts execution as soon as it receives the task. This process then incurs additional latency and overhead. After performing arithmetic operator fusion, the CPU emits fewer computational tasks to the GPU, effectively reducing the latency caused by task sending.

The completed result of an operator computation usually needs to be saved to the GPU's memory, and then the result of the previous operator is read out from the GPU's memory for computation when the next operator is executed. Operator fusion allows the intermediate results of several fused operators to be stored in registers, thus reducing the number of memory accesses to GPU memory. In the quantization of this paper, we fuse all activation operators with the previous operators, thus reducing the inference time.

In summary, the compression of the model is accomplished by using three methods simultaneously in the quantization process, taking into account the computational process of the hardware execution unit and the characteristics of SAR images.

3.2. Precision-Aware Scheduling

In the overall optimization of the network, the quantization of some layers will lead to a larger error in the whole neural network and eventually lead to a larger loss of accuracy. In order to reduce the accuracy loss due to model speed-up based on the previous steps, we fully analyze the computing power of the GPU platform and use the computational graph operator scheduling to ensure that the network does not lose accuracy after quantization. The accuracy loss of the network mainly comes from low-bit quantization. Mapping values from a high-precision format (FP32) to a low-precision format (INT8) will inevitably reduce the amount of information in the data, thus generating errors. The GPU platform we use has both INT8 and FP32 computing ability. We can use the high-precision FP32 format to calculate and store the layers with serious error loss and use the INT8 format to calculate and store the layers with less serious error loss. However, if too many layers are computed in FP32 form, although the loss of accuracy is lower, it also means that the detection will be slower at the same time, thus failing to maximize acceleration. After quantization, determining the number of layers for FP32 mode computation and storage is the key to solving the problem.

To determine the number of layers, we first constructed the optimization model. In the Yolo family of detection networks, the evaluation metrics describing the algorithm's detection accuracy are precision, recall, and mean accuracy (*mAP*). Among them, *mAP* is subdivided into *mAP*@0.5 and *mAP*@0.5:0.95. *mAP*@0.5 stands for the mean value of APs for each category when the Intersection over Union (*IoU*) threshold is 0.5. *mAP*@0.5:0.95 is the average of the APs for each category when *IoU* threshold takes 10 numbers increasing

from 0.5 to 0.95 in steps of 0.05, respectively. The formulae for the above metrics are as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

$$AP = \int_0^1 P(R)dR \tag{7}$$

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP \tag{8}$$

$$mAP@0.5 = \frac{1}{N}\sum_{i=1}^{N} AP_i(IOU_{\text{thresh}} = 0.5) \tag{9}$$

$$mAP@0.5:0.95 = \frac{1}{N}\sum_{i=1}^{N}\sum_{j} AP_i(IOU_{\text{thresh}} = j) \tag{10}$$

In the above formulae, TP represents the number of samples in which both the predicted and true values are positive samples. FP stands for the number of samples where the predicted value is positive but the true value is negative. FN stands for the number of samples where the predicted value is negative but the true value is positive. N denotes the number of categories detected.

Frames per second (*FPS*) is widely used as an indicator of the model detection speed, which is calculated as follows:

$$FPS = \frac{1}{t_{preprocess} + t_{\text{inf}ernce} + t_{NMS}} \tag{11}$$

The $t_{preprocess}$, $t_{\text{inference}}$, and t_{NMS} in the formula denotes the time consumed by pre-processing, inference, and Non-Maximum Suppression (NMS), respectively. In real heterogeneous computing systems, only the inference phase is run on the GPU platform as the usual case. Therefore, we choose the time length of inference as a metric to evaluate the speed of model detection.

Higher values of *mAP*@0.5 and *mAP*@0.5:0.95 mean higher detection accuracy of the model, and smaller $t_{\text{inference}}$ means faster detection. Therefore, our final optimization model is as follows:

$$\begin{gathered} \max mAP@0.5 = \frac{1}{N}\sum_{i=1}^{N} AP_i(IOU_{\text{thresh}} = 0.5) \\ \max mAP@0.5:0.95 = \frac{1}{N}\sum_{i=1}^{N}\sum_{j} AP_i(IOU_{\text{thresh}} = j) \\ \min t_{\text{inference}} \end{gathered} \tag{12}$$

There is a conflict among the three optimization objectives mentioned in Equation (12). In order to achieve the fastest speed within the error tolerance for each network, this paper proposes an accuracy-adaptive scheduling method to find the optimal number of scheduling layers. The method is divided into two parts. In the first part, we design the accuracy error detector and find the Pareto optimal set of solutions that fit in the optimization mode based on the results of the detection. In the second part, we propose the AHP-TOPSIS method to find the optimal compromise.

In order to obtain the most realistic Pareto optimal solution set, this paper first designs the error detector, which is planted after each quantized layer. Then, the images of the training set are selected for detection. During the detection process, the error detector records the output feature map produced by the quantized layer and calculates the MSE

between the value of the produced feature map and the value of the produced feature before optimization. For each network, we selected the layers with error values in the top 5%, starting with the layer with the largest error, and incrementally converted these quantized layers layer-by-layer into FP32 format for detection, and recorded the detection accuracy metrics *mAP*@0.5 and *mAP*@0.5:0.95 after each detection, as well as the time used for inference. These data are the set of solutions to the above problem.

In addition, in order to obtain the final solution, it is necessary to filter the obtained Pareto optimal solution set. In this paper, the multi-attribute decision-making method is introduced to achieve the above objectives. Therefore, this paper combines TOPSIS for multi-attribute decision making and also determines the weight values of each attribute by the analytic hierarchy process (AHP) [33].

The specific steps are shown below:

1. Insert the designed error detector into the detected network.
2. The error detector is used to calculate the error in the feature map values before and after the quantization of each layer.
3. Sort to select the top 5% of layers in the network with the largest errors.
4. The selected layers are incremented layer by layer for the inverse quantization operation.
5. The accuracy metrics and speed of detection are counted sequentially for each time as a Pareto solution set.
6. The maximum value of the inference time metric was subtracted from the corresponding solution in the set of Pareto-optimal solutions to standardize all metrics to a larger value for better results. The normalization matrix is also normalized so that the scale of the three metrics does not affect the results. Assume that the forwarding matrix is written as $Y = (y_{ij})_{m\times 3}$. The standardized matrix is as follows:

$$\begin{gathered} \boldsymbol{Z} = (z_{ij})_{m\times 3} \\ z_{ij} = \frac{y_{ij}}{\sqrt{\sum_{i=1}^{m} y_{ij}^2}}; i = 1, 2, \cdots, m; j = 1, 2, 3, 4 \end{gathered} \tag{13}$$

where m denotes the number of solutions in the set of Pareto solutions.

7. Construct the judgment matrix $\boldsymbol{A}$. The three optimization objectives, *mAP*@0.5, *mAP*@0.5:0.95, and $t_{\text{inference}}$ are analyzed two by two, and their importance is judged by the 1~9 scale method [34]. The judgement matrix takes the value of

$$\boldsymbol{A} = \begin{pmatrix} 1 & \omega_1 & \omega_2 \\ 1/\omega_1 & 1 & 1/\omega_3 \\ 1/\omega_2 & \omega_3 & 1 \end{pmatrix} \tag{14}$$

where ω_1 denotes the importance of *mAP*@0.5 with respect to *mAP*@0.5:0.95, ω_2 denotes the importance of *mAP*@0.5 with respect to $t_{\text{inference}}$, and ω_3 denotes the importance of *mAP*@0.5:0.95 with respect to $t_{\text{inference}}$. In this paper, the value of ω_1 is 9, the value of ω_2 is 7, and the value of ω_3 is 2.

8. Consistency indicators C_I and consistency ratios C_R are calculated through the maximum eigenvalue of the matrix $\lambda_{\max}$.

$$C_I = \frac{\lambda_{\max} - n}{n - 1} \tag{15}$$

$$C_R = \frac{C_I}{R_I} \tag{16}$$

In the formula, the matrix dimension $n = 3$ and the average random consistency index $R_I = 0.58$. When $C_R < 0.1$, the judgment matrix is considered to pass the consistency test, and the eigenvectors corresponding to the largest eigenvalues of the judgement matrix are

normalized to obtain the weights w_j. The weighted normalization matrix is calculated as follows:

$$\begin{gathered} V = (v_{ij})m \times 4 \\ v_{ij} = w_j Z_{ij}; i = 1, 2, \cdots, m; j = 1, 2, 3, 4 \end{gathered} \tag{17}$$

9. To calculate the relative approximation, the ideal and negative ideal solutions are first determined as

$$\begin{aligned} \boldsymbol{V}^+ &= \left\{ \left(\max_{1 \le i \le m} v_{ij} | j \in \boldsymbol{J}^+ \right), \left(\min_{1 \le i \le m} v_{ij} | j \in \boldsymbol{J}^- \right) \right\} \\ &= \left(v_1^+, v_2^+, \cdots, v_m^+ \right) \end{aligned} \tag{18}$$

$$\begin{aligned} \boldsymbol{V}^- &= \left\{ \left(\min_{1 \le i \le m} v_{ij} | j \in \boldsymbol{J}^+ \right), \left(\max_{1 \le i \le m} v_{ij} | j \in \boldsymbol{J}^- \right) \right\} \\ &= \left(v_1^-, v_2^-, \cdots, v_m^- \right) \end{aligned} \tag{19}$$

In the formula, $\boldsymbol{J}^+$ is the set of benefit-based attributes and $\boldsymbol{J}^-$ is the set of cost-based attributes. Then, the distance to the ideal and negative ideal solutions for each scenario can be calculated as:

$$D_i^+ = \sqrt{\sum_{j=1}^{3} \left(v_{ij} - v_j^+ \right)^2}; i = 1, 2, 3, \cdots, m \tag{20}$$

$$D_i^- = \sqrt{\sum_{j=1}^{3} \left(v_{ij} - v_j^- \right)^2}; i = 1, 2, 3, \cdots, m \tag{21}$$

This gives the relative approximation for calculating the scenarios as:

$$C_i = \frac{D_i^-}{D_i^+ + D_i^-}; i = 1, 2, 3, \cdots, m \tag{22}$$

10. The Pareto solution corresponding to the scheme with the highest relative approximation is chosen as the final result.

4. Results

In order to simulate the operation on a real airborne or onboard real-time processing platform, we use different computing platforms for training and testing. The training platform is the inspur high-performance workstation P8000 and the test platform is the NVIDIA Jetson AGX Xavier. A high-precision model is first trained on the training platform using the dataset and the selected model, and then this model is transferred to the test platform for the corresponding model compression, and finally the performance is tested on the test platform.

4.1. Experimental Platform

4.1.1. Training Platform

The training platform used in this paper is a high-performance platform with two Intel Xeon E5-2640 CPUs, 64GB DDR4 RAM, and two NVIDIA Quadro P5000 GPUs with 2560 CUDA cores and 16GB RAM. On top of the platform, we use the Ubuntu 20.04 LTS operating system and installed Python 3.8, Pytorch 1.11.0, cuda 11.3, and cudnn 8.6.0, which together constitute the software environment of the training platform.

4.1.2. Testing Platform

The test platform is the NVIDIA Jetson AGX Xavier development board, a high-performance embedded computing platform with 32 GB DDR4 memory, 512 CUDA cores, 64 Tensor cores, 8 CPU cores, and 2 NVDLA deep learning accelerators, which can provide

up to 30TOPS of deep learning computing power. Most notably, the NVIDIA Jetson AGX Xavier development board has significant advantages in terms of low power consumption and can fully meet the power consumption requirements of airborne or onboard edge processing platforms. For the software environment, we used the official NVIDIA JetPack SDK5.1.1, Pytorch1.12, cuda11.4, and cudnn8.6.0.

4.2. Dataset

The dataset used in this paper is the High-Resolution SAR Images Dataset (HRSID), which has been widely adopted by researchers and published by the University of Electronic Science and Technology. The dataset contains 5604 images and 16,965 ship targets. The images are of high quality from Sentinel-1B, TerraSAR-X, and TanDEM satellites and cover HH, HV, VV, and other polarization methods. In this article, we set the resolution of the image to 256 $\times$ 256 in order to obtain quick results.

4.3. Model

To prove the effectiveness of the approach in this paper, we have selected four different models from the yolov5 series, yolov5n, yolov5s, yolov5m, and yolov5l for our experiments.

Yolov5n: This is a variant of the Yolov5 series optimized for Nano devices such as the NVIDIA Jetson Nano. Yolov5n provides accuracy for edge devices while maintaining a faster speed.

Yolov5s: This is the smallest model in the Yolov5 family. This model performs better on devices with limited computational resources, such as mobile devices or edge devices. Yolov5s has a faster detection speed but relatively lower accuracy.

Yolov5m: This is a medium-sized model in the Yolov5 family. Yolov5m provides a better balance between speed and accuracy for devices with some computational power.

Yolov5l: This is one of the larger models in the YOLOv5 series. Yolov5l has relatively high accuracy but is slow to detect. It is suitable for tasks that require a high degree of accuracy.

The relevant parameters of the three models are shown in Table 1.

Table 1. Relevant parameters of the experimental model.

Model	Layers	Parameters	Flops
Yolov5n	157	1,760,518	4.1 g
Yolov5s	157	7,012,822	15.8 g
Yolov5m	212	20,852,934	47.9 g
Yolov5l	267	46,108,278	107.6 g

4.4. Result

In this paper, the effectiveness of the proposed speed-up method is evaluated in terms of both detection speed and detection accuracy. For detection accuracy, we use precision (P), recall (R), and mean average precision (*mAP*), which are commonly used in detection networks, as the evaluation metrics for detection accuracy. For detection speed, we use *FPS* as the evaluation index of detection speed, and which is calculated as Equation (12).

4.4.1. Effect of Each Method

In order to demonstrate the effectiveness of the methods in Section 3 for improving detection performance, we designed three ablation experiments. We used the two models mentioned in 4.2.1 without any added optimization as the baseline. The first experiment uses the method of SAR-aware model quantification based on baseline. The second experiment uses the method of precision-aware scheduling on the basis of the first experiment. The results of each optimization method are shown in Table 2. In Table 2, SAMQ denotes SAR-aware model quantification and PAS denotes Precision-aware scheduling.

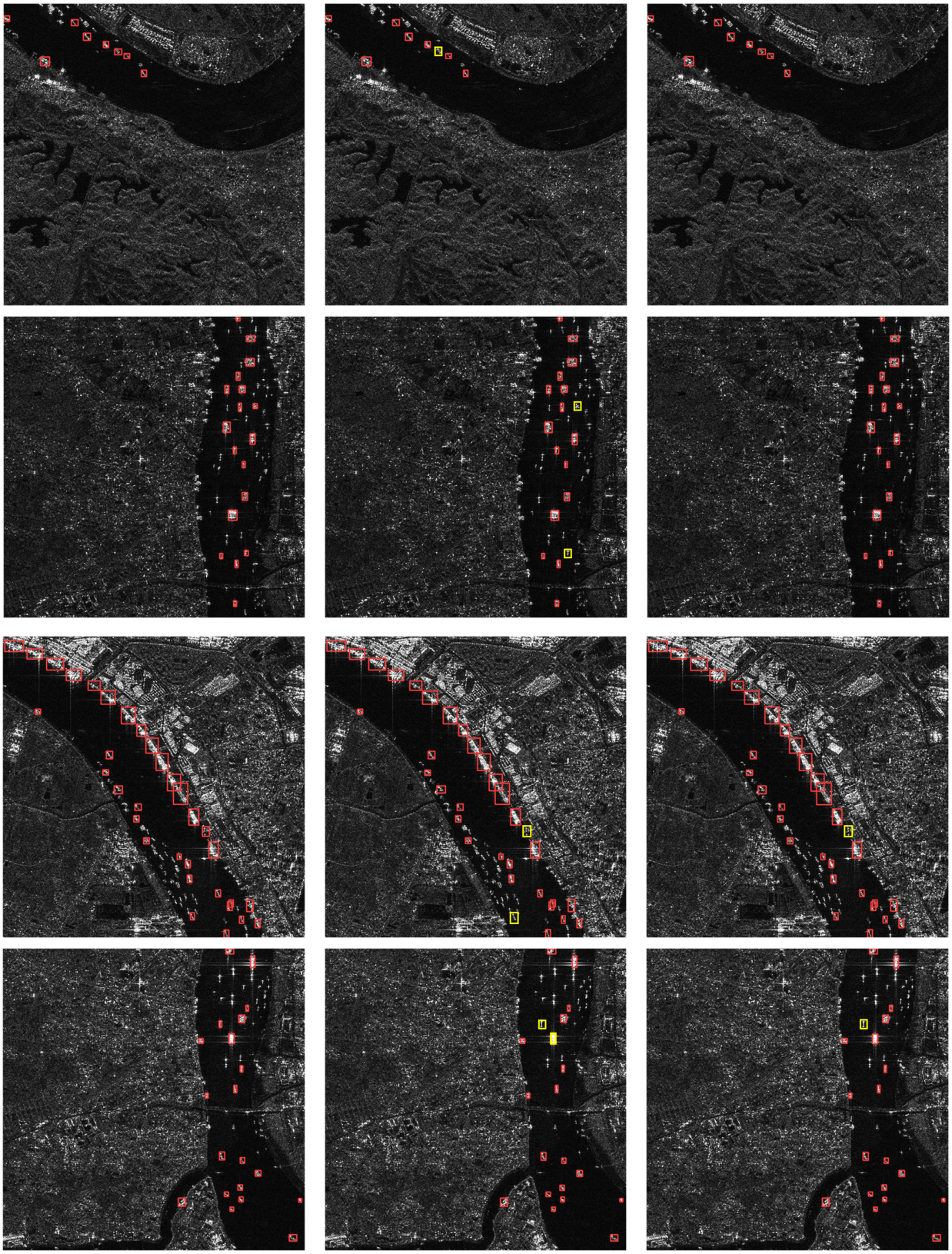
Table 2. Test performance comparison of models before and after using the method proposed in this paper.

Model	Method	P	R	*mAP*@0.5	*mAP*@0.5:0.95	*FPS*
Yolov5n	Baseline	0.8906	0.7141	0.7965	0.5227	190.8484
	SAMQ	0.8952	0.7123	0.7945	0.5077	237.5148
	SAMQ + PAS	0.8972	0.7145	0.7964	0.5155	234.7785
Yolov5s	Baseline	0.8973	0.7499	0.8237	0.5531	144.8165
	SAMQ	0.894	0.7506	0.8217	0.5438	218.2885
	SAMQ + PAS	0.8964	0.7516	0.8236	0.5438	212.8341
Yolov5m	Baseline	0.9098	0.7615	0.8364	0.5693	84.7072
	SAMQ	0.9012	0.7626	0.8289	0.5605	167.0161
	SAMQ + PAS	0.9096	0.7629	0.8363	0.5693	165.6523
Yolov5l	Baseline	0.9123	0.7723	0.8469	0.5772	57.1275
	SAMQ	0.8774	0.7699	0.8319	0.4605	140.4588
	SAMQ + PAS	0.9066	0.775	0.8467	0.5703	139.8758

Overall, the *FPS* of Yolov5n, Yolov5s, Yolov5m, and Yolov5l are improved by 43.9301, 68.0176, 80.9451, and 82.7483, after using the methods in this paper, while the reduction of *AP* is very small, decreasing by 0.0001, 0.0001, 0.0001, and 0.0002, respectively. From the above results, it can be demonstrated that the SAR ship target detection network using the method proposed in this paper achieves considerable speed with only a very low loss of accuracy on embedded GPUs.

We focus on analyzing the impact of each of the methods proposed in this paper on the results. Firstly, we analyzed the function of SAR-aware model quantification. According to the effect of SAR-aware model quantification, the *FPS* of Yolov5n, Yolov5s, Yolov5m, and Yolov5l are improved by 46.6664, 73.472, 82.3089, and 83.3313 after using the method of SAR-aware model quantification. The *AP* of yolov5n, yolov5s, and yolov5m are decreased by 0.002, 0.002, 0.0075, and 0.015, respectively, with yolov5l showing the most significant decrease in accuracy. It can be seen that the detection speed of the SAR ship target detection network is greatly improved by using this method, but there is still some loss of detection accuracy. Secondly, we analyzed the effect of precision-aware scheduling. After using the method of precision-aware scheduling, the *FPS* of Yolov5n, Yolov5s, Yolov5m, and Yolov5l are decreased by 2.7363, 5.4544, 1.3638, and 0.583. The *AP* of Yolov5n, Yolov5s, Yolov5m, and Yolov5l are improved by 0.0019, 0.0019, 0.0074, and 0.0148, respectively. Although this step gains a very low accuracy, the baseline is that after this step, the accuracy error is reduced to within 0.0001, and the loss of *FPS* is only in the single digits. This result achieves a perfect trade-off.

As can be seen from the detection results graph in Figure 5, after network inference acceleration using the SAMQ method, there are some missing targets in the detection results, which indicates that the detection accuracy of the accelerated model becomes lower. Using the PAS method on this basis, it can be seen that the missing targets are obviously reduced or have even disappeared. This shows the importance of the PAS method in reducing precision loss. Compared with the detection results before the acceleration method is used, there is only one or no missing targets, which proves that the accuracy loss of the proposed method can be ignored.



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

Figure 5. Comparison of detection results. From top to bottom are the results of yolov5n, yolov5s, yolov5m, and yolov5l model detection. Targets marked in red are detected. Targets marked in yellow are missed. (**a**) Baseline model detection results. (**b**) Detection results of the model after quantification using the SAMQ method. (**c**) Detection results of the model after using the PAS method with SAMQ.

4.4.2. Comparison Experiment

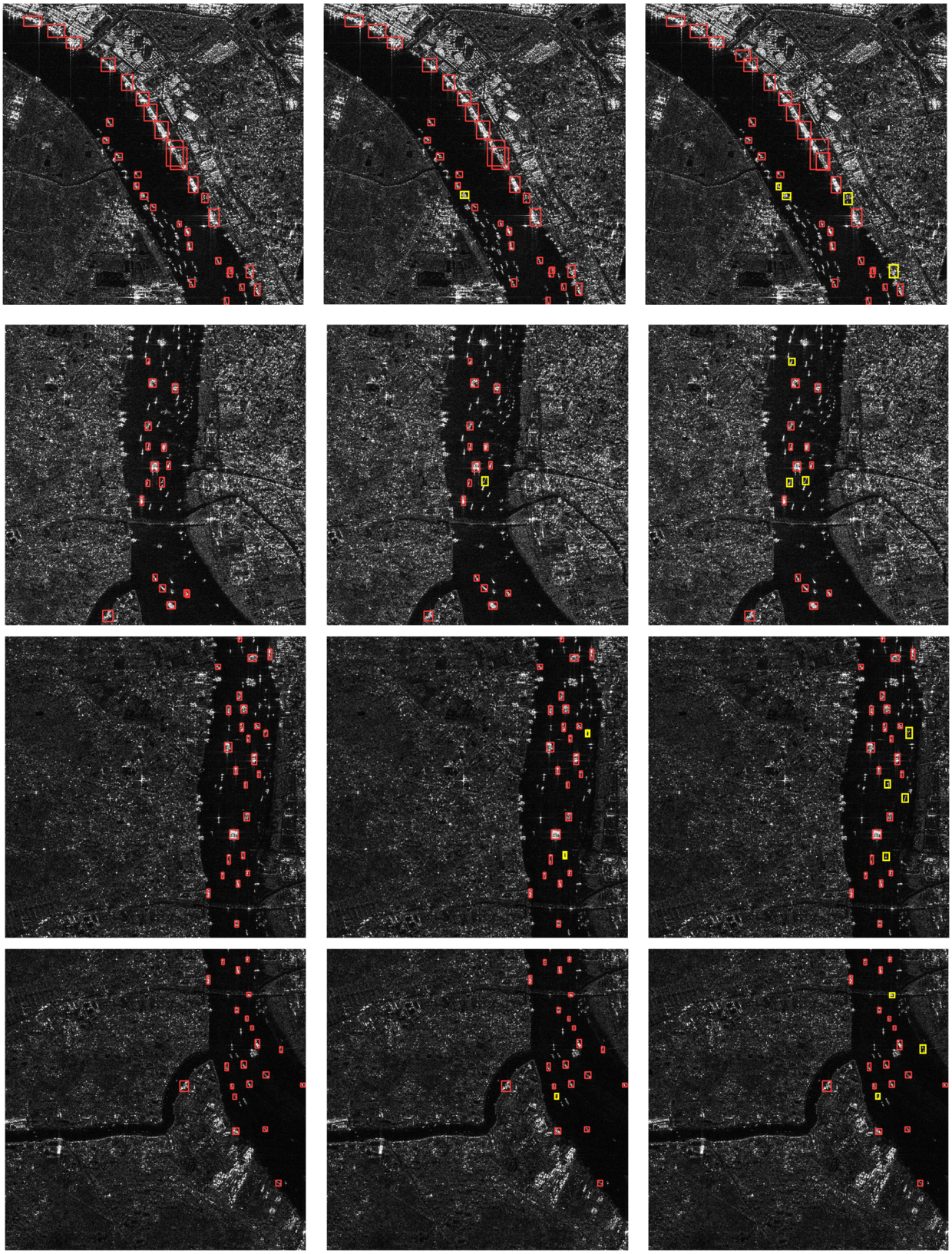
To verify the advancement of the speed-up method proposed in this paper, we accelerated four trained models using the speed-up method of the TensorRT proposed by NVIDIA, and we compared the performance of the speed-up results with those accelerated by the method proposed in this paper. The comparison results are shown in Table 3.

Table 3. Performance comparison results between the method proposed in this paper and TensorRT.

Model	Method	P	R	*AP*	*FPS*
Yolov5n	Baseline	0.8906	0.7141	0.7965	190.8484
	TensorRT	0.8771	0.7067	0.7896	235.6216
	SAMQ + PAS	0.8952	0.7123	0.7945	234.7785
Yolov5s	Baseline	0.8973	0.7499	0.8237	144.8165
	TensorRT	0.8789	0.7414	0.8171	212.2303
	SAMQ + PAS	0.8964	0.7516	0.8236	212.8341
Yolov5m	Baseline	0.9098	0.7615	0.8364	84.7072
	TensorRT	0.9028	0.7694	0.8352	93.2982
	SAMQ + PAS	0.9096	0.7629	0.8363	165.6523
Yolov5l	Baseline	0.9123	0.7723	0.8469	57.1275
	TensorRT	0.9069	0.774	0.846	60.8875
	SAMQ + PAS	0.9066	0.775	0.8467	139.8758

We analyze the results in Table 3 from the perspectives of both detection speed improvement and accuracy loss. Firstly, we compare the impact of TensorRT and the proposed method on the detection speed of SAR ship target detection networks. For small size networks such as Yolov5n and Yolov5s, the detection speeds after using TensorRT are 235.6216 and 212.8341. After using the method in this paper, the detection speeds are 234.7785 and 212.8341. The results of the two methods are basically the same. But for medium and large networks such as yolov5m and yolov5l, the TensorRT method gives very little improvement in the detection speed of the SAR target network for the detection network. The *FPS* is only improved by 8.591 and 3.76. After using our method, the *FPS* of these two networks is improved by 90.8268 and 82.7483, respectively. This result proves that our method is more effective in improving the detection speed for large-size networks. Secondly, the effect of the two methods on the loss of detection accuracy was compared. We found that the detection accuracy of SAR target network detection decreased to some extent after using TensorRT. Among them, the detection accuracy of Yolov5n, Yolov5s, Yolov5m, and Yolov5l has decreased respectively by 0.002, 0.002, 0.0012, and 0.0009. After using our method, these networks correspond to an accuracy loss of 0.0001, 0.0001, 0.0001, 0.0002. Compared to the loss of accuracy after using TensorRT, the loss incurred after using our proposed method is negligibly small. In summary, compared to TensorRT, the method proposed in this paper achieves a very low loss of accuracy while substantially increasing the detection speed, which is applicable to networks of various sizes.

As can be seen from the detection results in Figure 6, the models were detected after quantization using the method proposed in this paper and TensorRT, respectively. In this paper, we are more concerned with the comparison of detection results before and after using the acceleration method. Therefore, we directly used the unoptimized Yolov5 series of networks for SAR ship target detection and there are still some missed detections of ships in baseline. Compared to the detection results of the model without quantization, the detection results of the Yolov5m model after quantization using the proposed method produces two missed targets. The rest of the quantized models have only one missed target. However, the detection results of the models quantized using the TensorRT method all produced more than three missed targets.



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

Figure 6. Comparison of detection results. From top to bottom are the results of yolov5n, yolov5s, yolov5m, and yolov5l model detection. Targets marked in red are detected. Targets marked in yellow are missed. (**a**) Baseline model detection results. (**b**) Detection results of the model after quantification using the method proposed in this paper. (**c**) Detection results of the model after using TensorRT.

5. Discussion

After analyzing the results in Table 2, it can be seen that the proposed method in this paper has an overall significant speed-up for the SAR ship target detection algorithm, with 23.01%, 46.97%, 95.56%, and 144.85% improvement on the Yolov5n, Yolov5s, Yolov5m, and Yolov5l models, respectively, with negligible loss in accuracy.

Through careful analysis of each method proposed in this paper, we observe that all of them yield improvements. The extent of these gains varies across different networks due to their unique structures and computational volumes. For the method of SAR-aware model quantification, the improvements on Yolov5m and Yolov5l are 97.17% and 145.86%, while those on Yolov5n and Yolov5s are only 23.46% and 50.73%. Compared to lightweight networks, networks with more model layers tend to be more computationally intensive, so the speedup after quantization is more significant, and the corresponding accuracy loss is correspondingly larger. For different networks, the method of precision-aware scheduling reduces the accuracy loss before and after algorithm acceleration to within 0.0002 at the expense of negligible *FPS* performance, which demonstrates the compatibility of this method for each network. It is also clear that large networks have a greater demand to use this method to reduce the accuracy error before and after quantization.

In terms of the comparison results, TensorRT tends to produce good accelerations more easily on small networks, while the corresponding effect is little on large networks. After accelerating the neural network of lightweight type using this method, the performance of *FPS* is audible and obvious. However, for the detection accuracy of the model after using this method, there is still a certain degree of loss of accuracy. Compared with the method proposed in this paper, the accuracy loss of TensorRT is smaller, and the improvement of *FPS* is almost the same as that of TensorRT. For detection networks with larger models, the performance improvement by using TensorRT in *FPS* is very small, despite the low accuracy loss. The method proposed in this paper has a much lower accuracy loss and a much higher performance improvement in *FPS*. This proves that the method proposed in this paper outperforms TensorRT both in terms of loss of control accuracy and speedup and is more applicable to networks of various sizes.

6. Conclusions

In order to solve the problem of real-time SAR ship target detection algorithms on airborne or onboard processing platforms, a hardware-aware speed-up algorithm for SAR ship target detection networks is proposed in this paper. The proposed algorithm is divided into two parts: SAR-aware model quantification and Precision-aware scheduling. The method of hardware-aware model quantification uses low-bit quantization and operator fusion to reduce the time spent on computing and memory readings during operator computation, which accelerates inference operations. In addition, the loss of detection accuracy can be reduced using SAR-adapting calibration. The method of precision-aware scheduling, which de-quantifies the layers with the highest loss, allocates the computational resources on the available hardware in a rational way, reducing the loss of precision due to quantization. The algorithm is also tested on four models of the Yolov5 series, and the accelerated detection model is finally deployed on the NVIDIA AGX Xavier platform to demonstrate the effectiveness of the speed-up algorithm. This method has demonstrated superiority over the TensorRT method in terms of both detection accuracy and speed.

Author Contributions: Conceptualization, Y.Z.; Methodology, Z.Y.; Software, Y.Z. and C.L.; Validation, Y.Z. and Y.C.; Resources, J.X.; Data curation, Y.C.; Writing—original draft, Y.Z.; Writing—review & editing, S.J., J.X., X.Z. and Z.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

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

References

1. Yu, W.; Wang, Z.; Li, J.; Luo, Y.; Yu, Z. A Lightweight Network Based on One-Level Feature for Ship Detection in SAR Images. *Remote Sens.* **2022**, *14*, 3321. [CrossRef]
2. Zhang, J.; Xing, M.; Xie, Y. FEC: A Feature Fusion Framework for SAR Target Recognition Based on Electromagnetic Scattering Features and Deep CNN Features. *IEEE Trans. Geosci. Remote Sens.* **2021**, *59*, 2174–2187. [CrossRef]
3. Yang, M.; Guo, C. Ship Detection in SAR Images Based on Lognormal β-Metric. *IEEE Geosci. Remote Sens. Lett.* **2018**, *15*, 1372–1376. [CrossRef]
4. Leng, X.; Ji, K.; Zhou, S.; Zou, H. Noncircularity Parameters and Their Potential in Ship Detection from High Resolution SAR Imagery. In Proceedings of the 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Fort Worth, TX, USA, 23–28 July 2017; pp. 1876–1879.
5. Copeland, A.C.; Ravichandran, G.; Trivedi, M.M. Localized Radon Transform-Based Detection of Ship Wakes in SAR Images. *IEEE Trans. Geosci. Remote Sens.* **1995**, *33*, 35–45. [CrossRef]
6. Leng, X.; Ji, K.; Yang, K.; Zou, H. A Bilateral CFAR Algorithm for Ship Detection in SAR Images. *IEEE Geosci. Remote Sens. Lett.* **2015**, *12*, 1536–1540. [CrossRef]
7. Jiaqiu, A.; Xiangyang, Q.; Weidong, Y.; Yunkai, D.; Fan, L.; Li, S.; Yafei, J. A Novel Ship Wake CFAR Detection Algorithm Based on SCR Enhancement and Normalized Hough Transform. *IEEE Geosci. Remote Sens. Lett.* **2011**, *8*, 681–685. [CrossRef]
8. Jiao, J.; Zhang, Y.; Sun, H.; Yang, X.; Gao, X.; Hong, W.; Fu, K.; Sun, X. A Densely Connected End-to-End Neural Network for Multiscale and Multiscene SAR Ship Detection. *IEEE Access* **2018**, *6*, 20881–20892. [CrossRef]
9. Nie, X.; Duan, M.; Ding, H.; Hu, B.; Wong, E.K. Attention Mask R-CNN for Ship Detection and Segmentation from Remote Sensing Images. *IEEE Access* **2020**, *8*, 9325–9334. [CrossRef]
10. Zhao, Y.; Zhao, L.; Xiong, B.; Kuang, G. Attention Receptive Pyramid Network for Ship Detection in SAR Images. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2020**, *13*, 2738–2756. [CrossRef]
11. Cui, Z.; Li, Q.; Cao, Z.; Liu, N. Dense Attention Pyramid Networks for Multi-Scale Ship Detection in SAR Images. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 8983–8997. [CrossRef]
12. Zhang, T.; Zhang, X.; Shi, J.; Wei, S.; Wang, J.; Li, J. Balanced Feature Pyramid Network for Ship Detection in Synthetic Aperture Radar Images. In Proceedings of the 2020 IEEE Radar Conference (RadarConf20), Florence, Italy, 21 September 2020; pp. 1–5.
13. Lv, Z.; Lu, J.; Wang, Q.; Guo, Z.; Li, N. ESP-LRSMD: A Two-Step Detector for Ship Detection Using SLC SAR Imagery. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 1–16. [CrossRef]
14. Ai, J.; Tian, R.; Luo, Q.; Jin, J.; Tang, B. Multi-Scale Rotation-Invariant Haar-Like Feature Integrated CNN-Based Ship Detection Algorithm of Multiple-Target Environment in SAR Imagery. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 10070–10087. [CrossRef]
15. Li, J.; Qu, C.; Shao, J. Ship Detection in SAR Images Based on an Improved Faster R-CNN. In Proceedings of the 2017 SAR in Big Data Era: Models, Methods and Applications (BIGSARDATA), Beijing, China, 13–14 November 2017; pp. 1–6.
16. Yang, X.; Zhang, X.; Wang, N.; Gao, X. A Robust One-Stage Detector for Multiscale Ship Detection with Complex Background in Massive SAR Images. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 1–12. [CrossRef]
17. Ge, J.; Zhang, B.; Wang, C.; Xu, C.; Tian, Z.; Xu, L. Azimuth-Sensitive Object Detection in Sar Images Using Improved Yolo V5 Model. In Proceedings of the IGARSS 2022-2022 IEEE International Geoscience and Remote Sensing Symposium, Kuala Lumpur, Malaysia, 17 July 2022; pp. 2171–2174.
18. Chen, Y.; Duan, T.; Wang, C.; Zhang, Y.; Huang, M. End-to-End Ship Detection in SAR Images for Complex Scenes Based on Deep CNNs. *J. Sens.* **2021**, *2021*, 1–19. [CrossRef]
19. Zhu, M.; Hu, G.; Zhou, H.; Lu, C.; Zhang, Y.; Yue, S.; Li, Y. Rapid Ship Detection in SAR Images Based on YOLOv3. In Proceedings of the 2020 5th International Conference on Communication, Image and Signal Processing (CCISP), Chengdu, China, 13–15 November 2020; pp. 214–218.
20. Guo, Y.; Chen, S.; Zhan, R.; Wang, W.; Zhang, J. SAR Ship Detection Based on YOLOv5 Using CBAM and BiFPN. In Proceedings of the IGARSS 2022—2022 IEEE International Geoscience and Remote Sensing Symposium, Kuala Lumpur, Malaysia, 17 July 2022; pp. 2147–2150.
21. Cui, Z.; Wang, X.; Liu, N.; Cao, Z.; Yang, J. Ship Detection in Large-Scale SAR Images Via Spatial Shuffle-Group Enhance Attention. *IEEE Trans. Geosci. Remote Sens.* **2021**, *59*, 379–391. [CrossRef]
22. Zhou, L.-Q.; Piao, J.-C. A Lightweight YOLOv4 Based SAR Image Ship Detection. In Proceedings of the 2021 IEEE 4th International Conference on Computer and Communication Engineering Technology (CCET), Beijing, China, 13 August 2021; pp. 28–31.
23. Jiang, J.; Fu, X.; Qin, R.; Wang, X.; Ma, Z. High-Speed Lightweight Ship Detection Algorithm Based on YOLO-V4 for Three-Channels RGB SAR Image. *Remote Sens.* **2021**, *13*, 1909. [CrossRef]
24. Zhou, Z.; Chen, J.; Huang, Z.; Lv, J.; Song, J.; Luo, H.; Wu, B.; Li, Y.; Diniz, P.S.R. HRLE-SARDet: A Lightweight SAR Target Detection Algorithm Based on Hybrid Representation Learning Enhancement. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 1–22. [CrossRef]
25. Zheng, X.; Feng, Y.; Shi, H.; Zhang, B.; Chen, L. Lightweight Convolutional Neural Network for False Alarm Elimination in SAR Ship Detection. In Proceedings of the IET International Radar Conference (IET IRC 2020), Virtual; Institution of Engineering and Technology: Hong Kong, China, 2021; pp. 287–291.

26. Long, Z.; Suyuan, W.; Zhongma, C.; Jiaqi, F.; Xiaoting, Y.; Wei, D. Lira-YOLO: A Lightweight Model for Ship Detection in Radar Images. *J. Syst. Eng. Electron.* **2020**, *31*, 950–956. [CrossRef]
27. Xu, X.; Zhang, X.; Zhang, T.; Shi, J.; Wei, S.; Li, J. On-Board Ship Detection in SAR Images Based on L-YOLO. In Proceedings of the 2022 IEEE Radar Conference (RadarConf22), New York City, NY, USA, 21 March 2022; pp. 1–5.
28. Zhang, J.; Yang, J.; Li, X.; Fan, Z.; He, Z.; Ding, D. SAR Ship Target Detection Based on Lightweight YOLOv5 in Complex Environment. In Proceedings of the 2022 Cross Strait Radio Science & Wireless Technology Conference (CSRSWTC), Haidian, China, 17 December 2022; pp. 1–3.
29. Ren, X.; Bai, Y.; Liu, G.; Zhang, P. YOLO-Lite: An Efficient Lightweight Network for SAR Ship Detection. *Remote Sens.* **2023**, *15*, 3771. [CrossRef]
30. Banner, R.; Nahshan, Y.; Hoffer, E.; Soudry, D. Post-Training 4-Bit Quantization of Convolution Networks for Rapid-Deployment. *arXiv* **2019**, arXiv:1810.05723.
31. Nagel, M.; Fournarakis, M.; Amjad, R.A.; Bondarenko, Y.; van Baalen, M.; Blankevoort, T. A White Paper on Neural Network Quantization. *arXiv* **2021**, arXiv:2106.08295.
32. Wu, H.; Judd, P.; Zhang, X.; Isaev, M.; Micikevicius, P. Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation. *arXiv* **2020**, arXiv:2004.09602.
33. Yaraghi, N.; Tabesh, P.; Guan, P.; Zhuang, J. Comparison of AHP and Monte Carlo AHP Under Different Levels of Uncertainty. *IEEE Trans. Eng. Manag.* **2015**, *62*, 122–132. [CrossRef]
34. Tompkins, M.; Iammartino, R.; Fossaceca, J. Multiattribute Framework for Requirements Elicitation in Phased Array Radar Systems. *IEEE Trans. Eng. Manag.* **2020**, *67*, 347–364. [CrossRef]