Algorithm/Hardware Co-Design for Real-Time On-Satellite CNN based Ship Detection in SAR Imagery

Geng Yang, Jie Lei, *Member, IEEE*, Weiying Xie, *Member, IEEE*, Zhenman Fang, *Member, IEEE*, Yunsong Li, Jiaxuan Wang, Xin Zhang

***Abstract*—Recently, the convolutional neural network (CNN) based approach for on-satellite ship detection in synthetic aperture radar (SAR) images has received increasing attention, since it does not rely on predefined imagery features and distributions that are required in conventional detection methods. To achieve a high detection accuracy, most of the existing CNN-based methods leverage complex off-the-shelf CNN models for optical imagery. Unfortunately, this usually leads to expensive computational cost, which is hard to process in real time using resource-constrained devices deployed in the harsh satellite environment.**

In this paper, we propose OSCAR-RT, the first end-to-end algorithm/hardware co-design framework for On-Satellite CNN based SAR ship detection, which can simultaneously produce an accurate and hardware-friendly CNN model and an ultra-efficient FPGA-based hardware accelerator that can be deployed on satellites. With the real-time on-satellite processing speed in mind, we start from a state-of-the-art compact CNN model for optical imagery. To eliminate the sharp decrease in the detection accuracy for SAR imagery, we analyze the discrepancy between the SAR domain and optical domain, and propose to adapt the model by adjusting the output feature size to better detect relatively smaller objects in SAR imagery. To improve the detection speed, we propose to develop a fully-pipelined inter-layer streaming accelerator architecture, where all the layers of the CNN model can be concurrently processed using on-chip FPGA resources. To achieve this architecture, we first propose a hardware-guided, progressive, and structural pruning strategy, which is guided by our modeled hardware metrics and applies state-of-the-art coarse-grained and fine-grained filter pruning, as well as mixed-precision quantization techniques. Moreover, to improve the reusability and portability of the hardware accelerator design, we develop a library of highly optimized CNN components in high-level synthesis, together with their performance and resource models. Finally, we map the pruned CNN model onto these hardware library components in a fully-pipelined inter-layer streaming fashion, by adjusting their parallelism factors to balance the execution of each layer and fit into the resource constraint. Experimental results using the adapted MobileNetV1, MobileNetV2, and SqueezeNet models on the widely used SAR ship detection dataset (SSDD) demonstrate the effectiveness of OSCAR-RT: for the MobileNetV1 model, it achieves an average precision of 94%, a detection speed of 652 frames per second on the Xilinx VC709 FPGA evaluation board, while consuming about 5.8W power.

***Index Terms*—SAR imagery, ship detection, CNN acceleration, hardware-guided pruning, algorithm/hardware co-design**

I. Introduction

With the continuous development of spaceborne satellites, such as Sentinel-1, TerraSAR-X and Gaofen-3, synthetic aperture radar (SAR) imagery with advanced active microwave sensors plays an increasingly pivotal role in marine reconnaissance and surveillance, due to its distinctive advantages of all-weather, all-day, vast extent and high-resolution features. As ships are the main ocean carriers, the accurate detection of them from SAR imagery is of great significance in the real military and civil applications, such as maritime traffic control, ship rescue and battlefield awareness [1].

To ensure safety, security and economic benefits, these SAR imagery based applications usually require aggressive processing speed while maintaining a high ship detection accuracy. Recently, on-satellite detection technology, which directly performs data collection and data analysis on the fly, has emerged as a promising solution to address this dilemma. Nonetheless, deploying a real-time end-to-end ship detection solution for satellites, from high-accuracy detection algorithm designs to ultra-efficient hardware implementations, also brings several challenges.

From the algorithm design perspective, conventional ship detection methods—such as threshold methods [2], statistical methods [3], and transformation methods [4]—typically extract a ship region from sea clutter through the pre-estimated distribution or manually defined features for specific scenarios. For instance, the well-known constant false alarm rate (CFAR) method [5] calculates the adaptive threshold based on the hypothesized background statistical distribution under a given false alarm rate and finds the ship target pixels within the local reference window through the obtained threshold. In addition, many improved CFAR versions [6]–[12] are proposed to accommodate specific complex circumstances, such as crowded harbors, busy shipping lines, and oil-spilled oceans. However, the high fitting precision of the background usually leads to the high computational complexity of the

This work was supported in part by the National Natural Science Foundation of China under Grant 62071360, Grant 6207010444, Grant 61801359, Grant 61571345, Grant 91538101, Grant 61501346, Grant 61502367, in part by the Young Talent fund of University Association for Science and Technology in Shaanxi of China under Grant 20190103, in part by the Special Financial Grant from the China Postdoctoral Science Foundation under Grant 2019T120878, in part by the 111 project under Grant B08038, in part by the Fundamental Research Funds for the Central Universities under Grant JB180104, in part by the Natural Science Basic Research Plan in Shaanxi Province of China under 2019JQ153, Grant 2016JQ6023, and Grant 2016JQ6018, in part by the General Financial Grant from the China Postdoctoral Science Foundation under Grant 2017M620440, in part by the Yangtse Rive Scholar Bonus Schemes under Grant CJT160102, in part by the Ten Thousand Talent Program. ($Corresponding\ authors: Jie\ Lei, Weiying\ Xie$)

G. Yang, J. Lei, W. Xie, Y. Li, J. Wang and X. Zhang are with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China (e-mail: gengyang@stu.xidian.edu.cn; jielei@mail.xidian.edu.cn; wyxie@xidian.edu.cn; ysli@mail.xidian.edu.cn; xinzhang_xd@163.com).

Z. Fang is with School of Engineering Science, Simon Fraser University, Burnaby, BC, Canada (e-mail: zhenman@sfu.ca)

assumed model, resulting in complicated parameter estimation. More importantly, these methods customized for the specific scene do not provide robust detection accuracy in more general scenarios.

Recently, benefiting from the accomplishments of deep learning in optical imagery, the CNN-based detection methods have drawn wide attention, since they are able to learn parameters independently and extract features automatically to adapt to a variety of scenes. However, when these accurate CNN models are migrated directly from optical imagery to SAR imagery, they lose their accuracy in ship detection due to the intrinsic characteristics of the SAR imagery, i.e., SAR images are presented as grayscale images and ship regions can be very small and densely clustered. Most prior studies improve the detection accuracy by supplementing meticulous feature extraction modules into CNN models [13]–[23]. This comes at the expense of high computational cost and prohibits efficient hardware deployment to achieve high detection speed.

From the perspective of hardware implementation and deployment, the harsh on-satellite environment imposes stringent requirements on the size, weight, power, and reliability of the equipped devices. Unfortunately, most of the existing CNN-based methods for ship detection are prototyped on power-hungry GPU platforms [13], [15], [16], [18]–[22], which are not suitable for on-satellite deployment. On the other hand, Field Programmable Gate Array (FPGA) is a better alternative for on-satellite deployment, thanks to its flexible customization, low power, high performance, and more importantly, ionizing-radiation tolerance [24]. To meet the high throughput and low latency requirements of on-satellite SAR ship detection, one challenge is how to maximize the efficiency of the FPGA-based accelerator design via appropriate architecture design, resource allocation, and parallelism tuning, while satisfying the high heterogeneity in layer types and sizes of CNN models.

Even more challenging, there could often exist a conflict between the effective algorithm design to achieve high detection accuracy and the efficient hardware design to achieve high computing performance. On one hand, as mentioned earlier, a CNN-based method with high detection accuracy may lead to high computational complexity and inhibits efficient hardware implementation. On the other hand, an efficient hardware implementation, e.g., binarized neural networks on FPGAs [25], [26], may lead to significant detection accuracy drop. In fact, some optimizations that could improve the software performance may not improve the hardware performance. For example, numerous pruning strategies are proposed to reduce the number of operations (OPs) and parameters with tolerable accuracy loss [27]–[29]. However, recent studies show that the reduction of the OPs and parameters does not always lead to the improvement in hardware performance [30]. In summary, it remains as a challenge to design an accurate yet hardware-efficient CNN model for on-satellite SAR ship detection.

To address the aforementioned challenges, in this paper, we propose a novel algorithm/hardware co-design framework called OSCAR-RT, which spans from accurate CNN model design to efficient hardware implementation on a target FPGA for on-satellite SAR ship detection. In summary, this paper makes the following contributions:

1. The first end-to-end algorithm/hardware co-design framework for on-satellite SAR ship detection, called OSCAR-RT, which comprises of SAR-aware CNN model adapting, hardware-guided progressive pruning, and fully-pipelined inter-layer streaming accelerator design.
2. A SAR-aware CNN model adapting technique that adapts a compact CNN model from optical imagery domain—via adjusting its output feature size—to better detect relatively smaller objects in SAR imagery. This is based on our analysis and observation that a CNN's ability to detect small objects in SAR imagery is closely and positively correlated with its output feature size. This simple technique turns out to be very effective: it improves the SAR ship detection accuracy from 48.7% (original MobileNetV2 [31]) to more than 95% (our adapted MobileNetV2).
3. A hardware-guided, progressive, and structural pruning strategy, which is guided by our modeled hardware metrics and applies state-of-the-art coarse-grained and fine-grained filter pruning, as well as mixed-precision quantization techniques. To guide our model pruning, we develop performance and resource utilization models for our hardware accelerator library, and automatically generate a balanced solution that meets the target latency through flexible trade-offs between detection accuracy and resource cost.
4. A fully-pipelined inter-layer streaming accelerator architecture, where we develop a library of highly optimized, reusable, and portable hardware accelerator components for common CNN layers in high-level synthesis, and a fast mapping strategy to map all (heterogeneous and pruned) CNN layers onto these pre-built hardware components to run concurrently using on-chip FPGA resources.
5. Experimental results that demonstrate the effectiveness of OSCAR-RT using three lightweight models including MobileNetV1 [32], MobileNetV2 [31] and SqueezeNet [33] on the widely used SAR ship detection dataset (SSDD): for the adapted MobileNetV1 model, OSCAR-RT achieves a high average precision of 94%, a high detection speed of 652 frames per second and a low power consumption of about 5.8W power on the Xilinx VC709 FPGA evaluation board.

The rest of this paper is organized as follows. Section II reviews related work, including algorithm designs for CNN-based SAR ship detection and hardware acceleration for CNNs on FPGAs. Section III presents our proposed OSCAR-RT framework in detail. Experimental evaluation is demonstrated in Section IV. Finally, Section V concludes this paper.

II. Related Work

A. Traditional CFAR-based Ship Detection Methods

Various traditional ship detection methods have been proposed in recent decades [2]–[5], [34]–[37]. Among these methods, CFAR detector is widely studied and equipped for satellites [38] because of its target-irrelevant characteristic. The primary step of standard CFAR is to accurately model the background clutter using different models such as Gaussian [37], log-normal [34], generalized Gamma models [35] and

mixed models [36]. For instance, Leng $et\ al.$ combined the spatial distribution based on kernel density estimation on the original intensity model to reduce the influence of bright but relatively discrete SAR ambiguity and sea clutter [37].

However, highly heterogeneous clusters and interfering targets in the complex scenes usually contaminate the well-designed distribution, resulting in biased parameter estimation and inaccurate probability density function (pdf). Therefore, multiple improved CFAR variants—based on optimal statistics, or truncated clutter statistics, or data censoring—have been proposed [6]–[12]. For instance, [10] and [11] adopted the adaptive-threshold-based cluster truncation strategies to remove the high-intensity outliers caused by interfering target pixels, the azimuth ambiguities, and the breakwater in the multiple target environment. Ai $et\ al.$ [12] trimmed both high-intensity and low-intensity (such as spilled oil) outliers through the adaptive bilateral threshold and calculated the pdf of sustained real clutter samples using log-normal distribution. However, these traditional methods are highly dependent on real background modeling with complex parameter estimation for specific scenes. Further, one method cannot show competitive detection performance in different scenes.

B. CNN-based Ship Detection Methods in SAR Imagery

Recently, with the publication of some comprehensive and standard SAR ship datasets, such as SSDD [13] and HRSID [14], there is a growing body of literature that demonstrates the great potential of CNN-based methods in detection accuracy compared with traditional methods. The mainstream CNN-based detection methods depend on the prevailing achievements of CNNs for computer vision with optical images, which can be mainly grouped into two categories: two-stage methods [13]–[17] and one-stage methods [18]–[22].

The two-stage methods first utilize the neural network to generate the Region of Interest (ROI) proposals, and then the detection results are obtained by reprocessing the two branches of the classification and regression network. For instance, on the basis of standard Faster-RCNN [39], Li $et\ al.$ [13] employed a variety of optimization strategies such as feature fusion, transfer learning and hard negative mining to improve the ship detection performance. Cui $et\ al.$ [15] constructed a dense attention pyramid network by embedding a convolutional block attention module (CBAM) in the original feature pyramid network (FPN) [40], thereby strengthening the ability to detect ships in inshore areas. Similarly, Zhao $et\ al.$ [16] proposed an improved FPN with the combination of a novel lateral connection. Although these methods can acquire high accuracy, their complex and redundant detection schemes slow down the processing speed, which are hard to be applied to real-time applications such as maritime disaster relief and emergency military decisions.

To alleviate this problem, researchers introduced the one-stage methods that combine classification and detection into the same network for SAR ship detection. Chang $et\ al.$ [19] first proposed a YOLOv2-reduced network by removing some redundant top layers from the original YOLOv2 [20]. Zhang $et\ al.$ [21] proposed a grid CNN structure based on deep separable convolution inspired by YOLOv2. The efficient network structure proposed by Chen $et\ al.$ [22] adopted Darknet-53 to extract shallow location and deep semantic features and then utilized a top-down pyramid with concatenation to complete multi-scale detection in complex scenes. However, even for one-stage methods, their processing speed is still lagging behind the real-time requirements.

Unfortunately, due to the late start of CNN-based SAR ship detection, most of the existing solutions are prototyped on the power-hungry GPU platforms, which are impractical to deploy in the hash satellite environment. In short, existing algorithm optimizations did not consider the hardware-friendly characteristics of real satellite applications at the beginning of the model design. Our work aims to leverage the advantages of the one-stage methods, and provide high detection accuracy and hardware efficiency through algorithm/hardware co-design.

C. Hardware Acceleration for CNNs on FPGAs

Compared to GPUs, FPGAs have lower computing latency, and lower power and energy consumption. While compared to ASICs, FPGAs have more flexible reconfiguration and faster development cycle. Moreover, FPGAs are ionizing-radiation tolerant, which makes FPGAs a great hardware platform to accelerate on-satellite SAR ship detection. Unfortunately, existing FPGA-based implementations for SAR ship detection mainly target the traditional detection methods [38], [41] since CNN-based SAR ship detection methods are relatively new.

Meanwhile, FPGA has indeed become one of the mainstream candidate platforms for CNN acceleration. Existing representative studies on FPGA acceleartion for CNNs can be divided into two major categories according to their CNN accelerator architecture design: time-multiplexing layer-sharing architecture [42]–[46] and fully-pipelined inter-layer streaming architecture [25], [26], [47], [48].

The time-multiplexing layer-sharing architecture implements a big, unified and shared compute unit for all layers of the same type, and is shared by all CNN layers through time-multiplexing, i.e., it executes one CNN layer at a time. For instance, Zhang $et\ al.$ [42], [43] used the roofline model [49] to obtain the optimal parameter configuration for each layer by balancing computation and communication. The polyhedral-based framework named PloySA proposed in [44] can automatically generate the high-performance systolic array architecture for a standard convolution (SC) layer with a specific kernel size. Subsequently, they further designed foldable architecture that includes SC, depth-wise convolution (DWC) and pooling layer to achieve 2D pose detection for multiple people in optical images [45]. In [46], HybridDNN introduced a hybrid convolution engine using the fast Winograd algorithm. However, it is hard for the time-multiplexing layer-sharing architecture to adapt to the high heterogeneity of layer types and sizes in modern CNN models, since all heterogeneous layers of the same type share the same computing unit.

On the other hand, the fully-pipelined inter-layer streaming architecture maps all the CNN layers onto the on-chip FPGA resources at the same time and executes all layers concurrently in a fully-pipelined and streaming fashion. For

example, in FINN [25], the end-to-end mapping of binarized neural networks (BNNs) onto FPGAs is implemented through parametric and configurable building blocks. However, the extreme binary quantization applied by FINN is not suitable for tasks that require better accuracies. The second version of FINN, FINN-R [26], extends the original one with support for mixed and arbitrary precision. However, the model with higher than 4-bit precision may not be well implemented due to the non-general thresholding approach for batch normalization and non-linear function. DNNbulider proposed by Zhang $et\ al.$ [47] can automatically produce a fully-pipelined inter-layer streaming architecture leveraging highly optimized RTL-based components. Yet, there is a lack of support for new CNN modules such as DWC (depth-wise convolution). The champion work of low power object detection of DAC2020-SDC [50] adopts a fully-pipelined inter-layer streaming architecture to deploy a VGG-like model on the Ultra96v2 FPGA board. However, as will be presented in subsection III-C, we find that its core unit for SC (standard convolution) is still not resource-efficient enough due to the mismatch computing rate of its internal components.

From the performance perspective, the fully-pipelined inter-layer streaming architecture can achieve lower latency and higher throughput than the time-multiplexing layer-sharing architecture via fine-grained per-layer instantiation and concurrent and streaming layer execution. Meanwhile, the fully-pipelined inter-layer streaming architecture also has a more stringent requirement on the model size: the CNN model size needs to be compact enough such that all layers can fit onto the FPGA on-chip resources.

In this paper, we intentionally start with a widely used compact CNN model, i.e., MobileNetV1 [32], MobileNetV2 [31] and SqueezeNet [33], and adapt it to the SAR imagery to improve its detection accuracy to more than 95%. Moreover, we apply hardware-guided pruning to further reduce the model size with less than 2.1% accuracy drop, and then implement a highly optimized fully-pipelined inter-layer streaming architecture on the FPGA. To the best of our knowledge, we are the first to propose an end-to-end algorithm/hardware co-design framework for on-satellite CNN-based SAR ship detection.

III. Our Proposed OSCAR-RT Framework

Fig. 1 depicts the overall design flow of OSCAR-RT that integrates SAR-aware model adapting, hardware-guided progressive pruning, and fully-pipelined inter-layer streaming accelerator mapping. It takes the target detection accuracy (Acc_{tar}) and latency (Lat_{tar}), as well as the target FPGA platform with the available resource (Res_{total}), as inputs. With the target latency in mind, we employ the widely used lightweight one-stage CNN models from the optical image domain as the initial reference model. To obtain a high-accuracy network structure suitable for SAR image domain, we carefully analyze the data characteristics of SAR images and propose to adapt the CNN model by adjusting its output feature map size to better detect small objects.

To fit the entire (adapted) CNN model onto the FPGA chip, we propose a hardware-guided progressive pruning strategy and a fully-pipelined inter-layer streaming accelerator architecture. To eliminate the drawbacks of existing pruning methods that mainly aim to reduce the number of parameters and operations [30], we use hardware metrics—including hardware parallelism and resource cost—to guide our structural pruning and progressively compress the model to generate a series of CNN candidates and their corresponding optimized hardware configurations, while satisfying the target accuracy. In our pruning strategy, we apply hardware-friendly coarse-grained and fine-grained filter pruning, as well as mixed-precision quantization (4-bit weights, and 3-bit to 6-bit activations), to maintain a less than 2% accuracy loss.

To improve the reusability and portability of our hardware accelerator design, we develop a library of highly optimized and configurable CNN components in high-level synthesis (HLS), as well as their performance and resource models. Based on this library, we map our pruned CNN model onto these hardware components in a fully-pipelined inter-layer streaming fashion, such that all CNN layers can run concurrently on the FPGA. During the mapping process, we adjust each layer's parallelism factor to balance the execution latency of all layers and fit all layers onto the on-chip FPGA resources.

The final output of our OSCAR-RT framework is a series of optimized CNN models (CNNs) and their corresponding optimized hardware designs (HW_Archs). The whole process is automated. Users can select one of the candidates as their final choice, based on the trade-off of the detection accuracy, latency, throughput, and resource utilization. Finally, to quickly verify the on-board performance of our generated model and hardware architecture, we provide users with a unified and automated CPU-FPGA heterogeneous testing system, which can realize the pipeline scheduling among the reading of input samples on CPU, major CNN computation on FPGA, and post-processing on CPU, to overlap their execution.

A. SAR-Aware Model Adapting

When a prevalent CNN network for the optical imagery is migrated to SAR ship detection tasks, it usually cannot achieve the best accuracy due to the essential difference of microwave imaging mechanism. For instance, the detection accuracy by applying the original reference MobileNetV2 model [31] to the widely used SSDD SAR dataset [13] is very low. As shown in Fig. 2, the average precision (AP) value obtained by the original MobileNetV2 model under different input image sizes ranges from 32.8% to 59%, accompanied by a large number of missed detections, false alarms and high positioning errors (will be presented in Fig. 11 and Fig. 12 in Section IV-B).

To understand the reason behind the accuracy degradation, we analyze the SAR images and find that the ship objects are much smaller and densely clustered in the grayscale SAR images. Therefore, we further characterize the relationship between the SAR ship detection accuracy and the output size of the feature extractor. As shown in Fig. 2, the output size of the feature extractor is positively correlated with the ability to detect small targets: the larger the output feature size, the higher the detection accuracy, since a larger output feature size preserves more object info from the image.

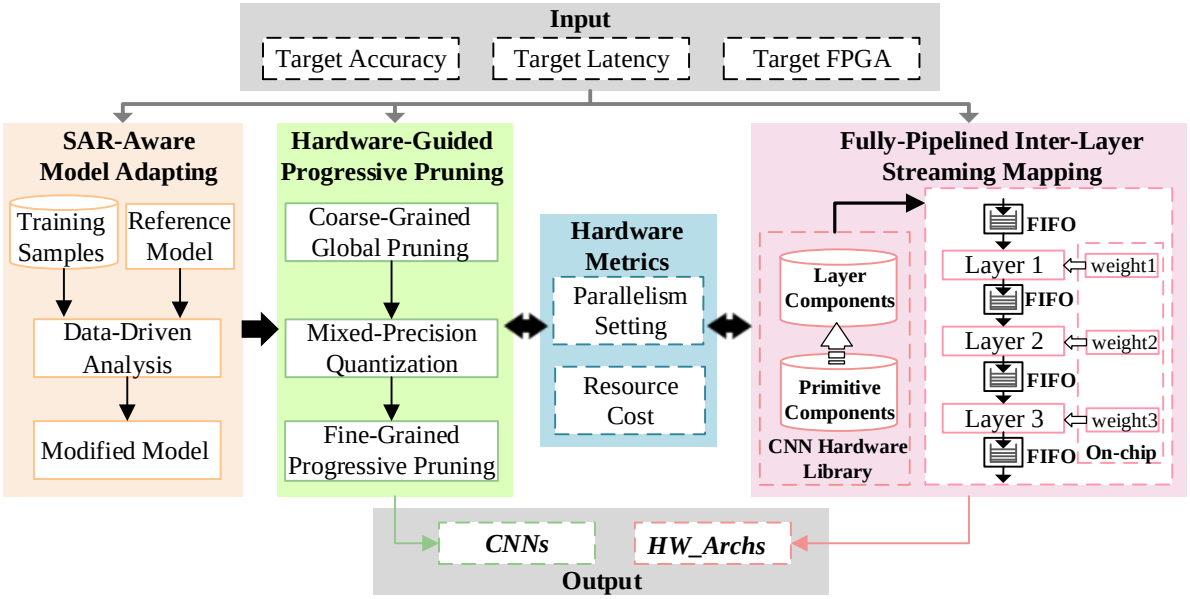


Fig. 1. An overview of our end-to-end algorithm/hardware co-design framework, OSCAR-RT, for on-satellite CNN-based SAR ship detection.

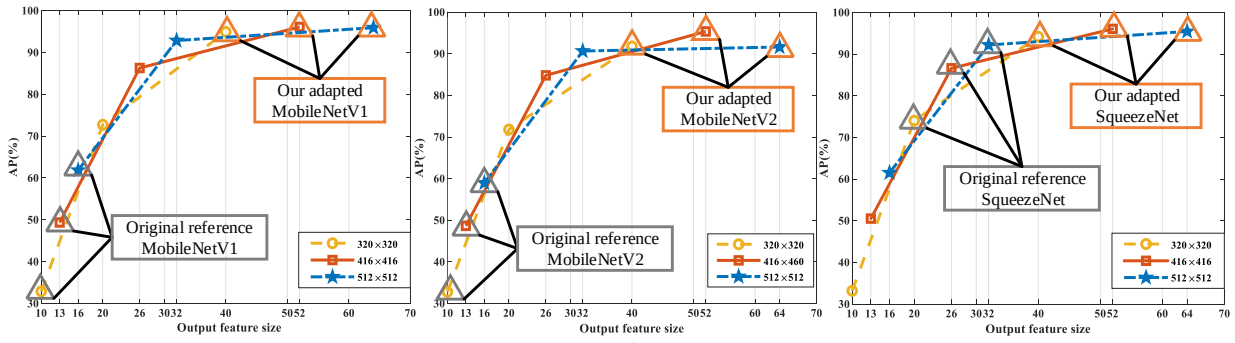


Fig. 2. The impact of output feature size on the detection accuracy over SSDD dataset. 320×320, 416×416 and 512×512 denote different input image sizes. The output feature sizes of 10, 13, 16 were used in the original MobileNetV1 and MobileNetV2 for the three input image sizes, respectively. The output feature sizes of 20, 26, 32 were used in the original SqueezeNet model for the three input image sizes, respectively. The output feature size is changed by only modifying the stride of the last convolutional operations or removing the max-pooling operations in the original models.

However, most of the networks used for optical images are initially designed for classification tasks that only require the deep semantic information hidden in the image. Therefore, these CNN models usually employ relatively many down-sampling operations (e.g., convolution layer with stride 2, max-pooling layer) to obtain small-scale output feature maps. When these networks are applied to SAR ship detection, the small-size target information with very few pixels will be gradually lost with the deepening network structure, resulting in unsatisfactory detection performance.

Based on this observation, we adapt the CNN model by setting an appropriate output feature size while maintaining the structural advantages of the original model. As a result, we can improve the SAR ship detection accuracy to more than 95.4% for 416×416 input image size. An in-depth evaluation is provided in Section IV-B. It is worthwhile to mention that the modification based on an existing lightweight model is more efficient and cost-effective than starting from scratch.

In addition, ab initio training strategy [51] is applied to reduce the learning objective bias caused by the pre-trained weights using optical images such as ImageNet. Finally, the modified CNN model with pre-trained weights is settled down. The layer-wise information, including layer type and layer size, is extracted and passed to the next step.

B. Hardware-Guided Progressive Pruning

Although the one-stage lightweight model (with our model adaption) is used as our initial reference, it is still impractical to host the entire CNN model with all parameters in all layers in resource-constrained FPGAs. Therefore, we propose a progressive and structural pruning strategy in Algorithm 1, which is guided by our modeled hardware metrics (in Section III-D) to iteratively prune the CNN layers that cost the maximum amount of hardware resources, under the constraints of the maximum accuracy loss ϵ, target latency Lat_{tar}, and available resource Res_{total}. To reduce the execution time of the iterative pruning, initially, we also apply a coarse-grained filter pruning to prune all layers globally and conservatively. Finally, we also apply state-of-the-art mixed-precision quantization [50], [52] to further reduce the model size. Next we present the coarse-grained and fine-grained filter pruning and mixed-precision

quantization aware training steps in Algorithm 1 in detail.

Algorithm 1: Pseudocode of hardware-guided progressive pruning

Input: CNN with D convolutional (including SC and DWC) layers; maximum accuracy loss: ϵ; target latency: Lat_{tar}; available resource: Res_{total}
Output: A set of CNN candidates: $\hat{CNN}$
Tunable Hyper Parameters: Total number of iterations: $iter$; global pruning rate: $rate$; filters to prune in fine-grained layer-wise pruning: f_k; quantization bit-width for weights and activations: W, A.

1: CNN_0 = GlobalFilterPruning(CNN, $rate$);
2: LongTermTraining(CNN_0);
3: MPQAT$\big(CNN_0, \{W_j, A_j\}_{j=1}^{D}\big)$;
4: TuneHardwareParallelism(CNN_0, Lat_{tar});
`// Tune hardware parallelism based on inter-layer matching (Section III-D)`
5: **for** j from 1 to D **do**:
6: $Res_{L_j}^{CNN_0}$ = GetResourceCost(CNN_0, j); `// Get resource cost for layer` L_j `with Eq.` (9)
7: **for** i from 1 to $iter$ **do**:
8: L_{max} = FindMaxResCost$\big(Res_{L_j}^{CNN_{i-1}}(j = 1..D)\big)$;
9: CNN_i = LayerFilterPruning(CNN_{i-1}, L_{max}, f_k);
10: ShortTermTraining(CNN_i);
11: **if** $Acc^{CNN_0} - Acc^{CNN_i} > \epsilon$ **do** break;
12: TuneHardwareParallelism(CNN_i, Lat_{tar});
13: **for** j from 1 to D **do**:
14: $Res_{L_j}^{CNN_i}$ = GetResourceCost(CNN_i, j);
15: **if** $\sum_{j=1}^{D} Res_{L_j}^{CNN_i} \leq Res_{total}$ **do**:
16: $CNN_i \in \hat{CNN}$;
17: return $\hat{CNN}$

1) Coarse-Grained Global Filter Pruning: To expedite the pruning process, the coarse-grained global filter pruning (lines 1-2 in Algorithm 1) is first performed to simultaneously remove a certain proportion ($rate$) of filters from all layers in the adapted CNN model, while maintaining the accuracy. According to [29], the average rank of the feature map generated by a single filter almost keeps constant and independent of the amount of input data, and the feature maps with higher ranks contribute more to the accuracy. Inspired by this observation, we first remove the filters corresponding to the feature maps with relatively lower ranks. To maintain negligible accuracy drop induced by this global pruning, we apply long-term training with a relatively large epoch range from 1K to 2K. In our experiment, the removal of more than 25% of the parameters in this stage brings less than 0.2% accuracy loss while the computational complexity reduced by more than 26%.

2) Mixed-Precision Quantization Aware Training: Although the operators represented by float32 can capture more details during the back propagation of the training process, the trained parameters are proved to contain a lot of redundant information in the inference stage [26]. Based on this, we subsequently apply state-of-the-art low-bit mixed-precision quantization aware training (MPQAT) [50], [52] to flexibly reduce the size of both weights and activations (line 3 in Algorithm 1). In order to reduce the search space while ensuring efficient hardware implementation, the different bit-widths (W_j for weights and A_j for activations) is only adopted for inter-layer, while the intra-layer remains unified precision. This can be efficiently implemented in our fully-pipelined inter-layer streaming accelerator design (in Section III-D). Our experimental results in Section IV show that our quantization strategy imposes a small impact on accuracy while greatly reducing the memory consumption and computational cost.

3) Fine-Grained Progressive Filter Pruning: After the coarse-grained global pruning and mixed-precision quantization, we apply the progressive filter pruning with the guidance of direct hardware metrics to further compress the model in a more fine-grained layer-wise fashion (lines 4-16), so as to obtain hardware-efficient CNN candidates suitable for the target FPGA. In each iteration, we find the layer with the highest resource cost and remove its last f_k filters in sequence. Subsequently, we fine-tune the updated network with short-term training (the training epoch ranges from 50 to 200) to restore its accuracy. If the accuracy loss is larger than ϵ, we stop the pruning algorithm; otherwise, we update the hardware parallelism factors and the resource cost of each layer according to the inter-layer streaming matching in Section III-D. If the total resource of all layers fits within the available resource Res_{total}, then this CNN model, together with its optimized hardware configuration, will be added to the candidate CNN set $\hat{CNN}$.

It should be noted that the concept of a layer can be extended to a collection of multiple layers to adapt to the CNN structure and accelerate the progressive pruning process. At the end of this algorithm, a sequence of CNN candidates from each iteration, together with their optimized hardware architecture configurations, are generated to providing users with a variety of deployment options through the trade-off between accuracy, resource cost, and throughput. In particular, if the target FPGA is replaced, users can directly select or fine-tune one of the CNN candidates without expensive overhead to redo the whole pruning process.

C. Portable and Configurable CNN Hardware Library

To improve the useability and portability of OSCAR-RT for different CNN models and different FPGAs, we build an HLS-based CNN hardware library, including primitive components and layer components with configurable parameters. To be specific, the primitive components consist of a series of functional logics such as same padding unit (SamePad), sliding window unit (SWU), Matrix-Vector-Activation unit for depth-wise convolution (D-MVAU),Matrix-Vector-Activation Unit for standard convolution (S-MVAU) and max-pooling unit(MPU). The layer components, such as SC (standard convolution), PWC (point-wise convolution, a special case as 1×1 SC), DWC (depth-wise convolution), the residual unit [53], and fully-connected layer, can be easily constructed through the pipeline sequence of those primitive components. Note that all of our hardware library components use the streaming interface for efficient processing and easy construction of a

```
1: for(or = 0; or < OR; or++){
2:     for(oc = 0; oc < OC; oc++){
3:         for(ouf = 0; ouf < M/Oup; ouf++){
4:             for(inf = 0; inf < N×K×K/Inp; inf++){
5: #pragma HLS PIPELINE II=1
6:                 bufI[Inp]←readInput(Inp,Oup);
                   //read Inp input pixels from IFM processed by SWU
7:                 for(o = 0; o < Oup; o++){
8: #pragma HLS UNROLL
9:                     bufW[Inp]←readWeight(Inp,Oup); //read Inp weights
10:                    for(i = 0; i < Inp; i++){
11: #pragma HLS UNROLL
12:                        partialSum+=bufI[i]×bufW[i];
                           //pixel-by-pixel multiply-accumate operation
13:                    }
14:                    finalSum[o]+=partialSum;
15:                    BufO[o]=ACTU(finalSum[o]);
16:            }}//end of inf loop
17:            writeOutput(BufO[Oup]);
18: }}}
```

Fig. 3. Pseudocode of the baseline S-MVAU with some inefficiency.

TABLE I. Design Variables and Explanations

Variables	Explanation
N	number of input channels of IFM
M	number of output channels of OFM
R/C	number of rows and columns of IFM
OR/OC	number of rows and columns of OFM
S	stride size
K	kernel size
P	padding size
(W, A)	bit-width of weight and activation
(Inp, Oup)	parallelism factor along input and output channels

user-defined CNN accelerator. Our proposed hardware library supports most of the existing CNN models.

Moreover, during the construction of the CNN hardware library, the latency and resource cost of each component can be evaluated in advance through pre-modeling without waiting for the entire CNN model and accelerator to be built. For the sake of simplicity, we only introduce the core primitive components of CNN, i.e., S-MVAU and D-MVAU, in detail.

1) S-MVAU Design and Modeling: As shown in Fig. 3, the standard convolution (SC) takes N input feature maps (IFM) with size of $R \times C$, and applies standard convolution with $M \times N$ filters (size of $K \times K$) to generate M output feature maps (OFM) with size of $OR \times OC$. In order to increase the utilization of computation resources in the FPGA, it is necessary to pipeline and/or unroll the deeply nested multiply-accumulate loops in the convolution operation along different dimensions to achieve a high degree of parallelism. Similar to [43] [54] [50], in this paper, we unroll for the input channel dimension of IFM and the output channel dimension of OFM, and pipeline the rest of the loop nests with a pipeline initiation interval of one (i.e., II=1). We define two parameters, the input parallelism (Inp) and the output parallelism (Oup), which can be flexibly configured by users when they use our S-MVAU component. All the configurable parameters of our hardware library components are summarized in Table I.

Driven by this unrolling mechanism, [54] and [50] adopted the baseline hardware structure of S-MVAU shown in Fig. 4(a). In specific, Inp pixels derived from the streaming IFM, which

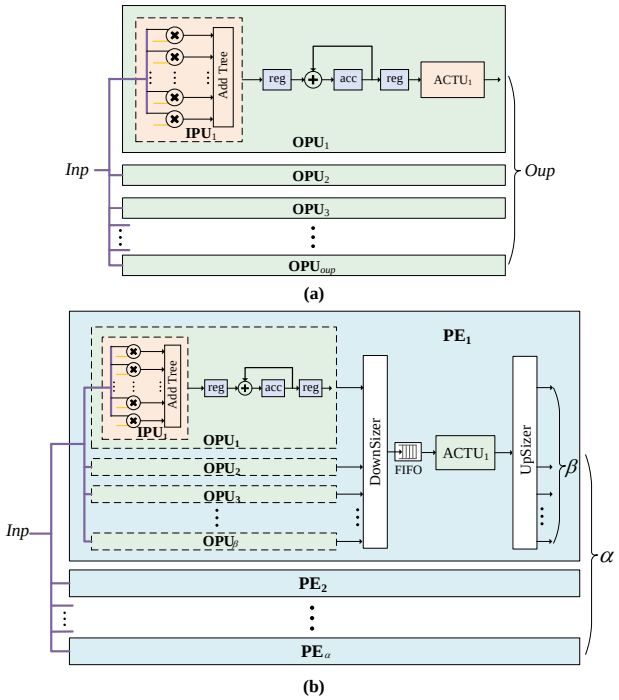


Fig. 4. (a) Inefficient baseline hardware structure of S-MVAU [50], [54]. (b) Our optimized hardware structure of S-MVAU.

is processed by SamePad and SWU units (SamePad and SWU are not required for 1×1 SC), are first fed simultaneously into Oup output parallel units (OPU). In each OPU, there is one input parallel unit (IPU) that performs Inp-way parallel multiply-accumulates of the input pixels and the corresponding weights. Considering the working mechanism of S-MVAU, the weight layout is reorganized offline to reduce the additional memory access overhead of noncontiguous weight data. To get the result for each output pixel, the IPU has to execute $\frac{N \times K \times K}{Inp}$ times; and each time, the partial sum from the IPU will be accumulated to get the final sum result. Finally, the activation unit (ACTU) merges the non-linear activation operation (e.g., ReLU) with the batch normalization, and applies to the final sum to obtain the final pixel for the OFM. Our FPGA implementation of the ACTU is similar to the one used in [50], which could result in a small tolerable accuracy loss, since the division operation in batch normalization is replaced with the right shifter to improve the hardware performance.

One deficiency with this baseline S-MVAU design is that, the IPU takes $\frac{N \times K \times K}{Inp}$ cycles to produce a final sum, whereas the ACTU could consume a final sum every cycle. Such mismatched data rate makes the ACTU in an idle state for most of the time. In addition, with the increase of Oup, more ACTUs also cause unnecessary resource overhead; note that the ACTU performs batch normalization as well and consumes notable amount resource. To address this issue, we propose an optimized S-MVAU hardware structure in Fig. 4(b), which consists of α identical processing engines (PEs) that work in parallel. In each PE, we add a DownSizer to perform the parallel-to-serial conversion to enable $\beta = \frac{N \times K \times K}{Inp}$ OPUs to share one ACTU. Finally, we add an Upsizer to restore

the serial resulting pixels processed by the ACTU to parallel pixels. Such optimized structure not only ensures that ACTUs are always working at full capacity but also reduces the resource overhead especially for large Oup. The number of PEs, α, can be calculated as:

$$\alpha = \frac{Oup}{\beta} = \frac{Oup \times Inp}{N \times K \times K} \tag{1}$$

Based on our proposed optimized structure, since the outer loops in Fig. 3 are pipelined with an II=1, the latency of S-MVAU is estimated as:

$$\begin{aligned} Lat_{S-MVAU} &= \frac{OR \times OC \times M \times N \times K \times K}{Inp \times Oup \times Freq} + \frac{PP}{Freq} \\ &\approx \frac{OR \times OC \times M \times N \times K \times K}{Inp \times Oup \times Freq} \end{aligned} \tag{2}$$

where

$$\begin{aligned} OR &= \lfloor \frac{R + 2 \times P - K}{S} + 1 \rfloor \\ OC &= \lfloor \frac{C + 2 \times P - K}{S} + 1 \rfloor \\ PP &= pipeline\_depth - 1 \end{aligned} \tag{3}$$

where $Freq$ is the clock frequency, and $Inp < K \times K \times N$. The configurable parameters involved are shown in Table I.

The resource cost is computed as:

$$\begin{aligned} Res_{S-MVAU} &= \alpha \times (\beta \times Res_{OPU} + Res_{ACTU}) + \Gamma \\ &= Oup \times Res_{OPU} + \alpha \times Res_{ACTU} + \Gamma \end{aligned} \tag{4}$$

where Γ represents the fixed resource overhead caused by the control logic. Res_{OPU} is the resource cost consumed by the OPU, which is directly related to Inp, W and A. Res_{ACTU} refers to the resource cost caused by the non-linear activation function that incorporates the BN operation in the ACTU. All these Γ, Res_{OPU}, and Res_{ACTU} resource cost can be obtained by the HLS synthesis report.

2) D-MVAU Design and Modeling: Depth-wise convolution (DWC), which is often followed by 1×1 SC (i.e., point-wise convolution), plays an important role in modern CNN designs [55]. Therefore, we supplement the corresponding hardware structure of DWC to enhance the versatility of our hardware library. Specifically, DWC applys the filter kernel to each input channel individually to generate the same number of output channels ($M = N$). Our hardware design of D-MVAU is depicted in Fig. 5. Inp pixels from the streaming IFM (processed by SamePad and SWU units) are divided into α pixel sets (each set contains $\beta = K \times K$ pixels), and these pixel sets are delivered to α PEs respectively. In each PE, β parallel units (PUs) execute $K \times K$ multiply-accumulates concurrently to generate β final sums. Subsequently, similar to S-MVAU, one ACTU is shared by these PUs to obtain the final pixels of the OFM in order. In particular, Inp and Oup are always equal for DWC due to its essential computational properties.

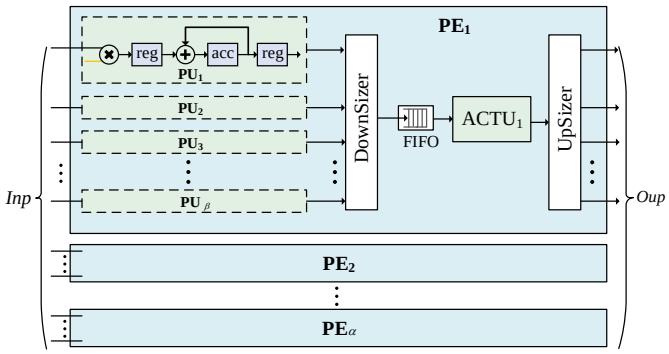


Fig. 5. Hardware structure of D-MVAU, where $M = N$ and $Oup = Inp$.

Similar to S-MVAU, the latency of D-MVAU is estimated as:

$$\begin{aligned} Lat_{D-MVAU} &= \frac{OR \times OC \times N \times K \times K}{Inp \times Freq} + \frac{PP}{Freq} \\ &\approx \frac{OR \times OC \times N \times K \times K}{Inp \times Freq} \end{aligned} \tag{5}$$

where $Inp < K \times K \times N$.

Its resource cost is computed as:

$$\begin{aligned} Res_{D-MVAU} &= \alpha \times (\beta \times Res_{PU} + Res_{ACTU}) + \Gamma \\ &= Inp \times Res_{PU} + \alpha \times Res_{ACTU} + \Gamma \end{aligned} \tag{6}$$

where Res_{PU} denotes the resource cost of each PU, which is related with W and A. The PU occupies a small amount of LUTs and FFs without using DSPs, as it only uses one low-bit-width multiplication and addition. Similarly, it can be obtained via the HLS synthesis report. The number of PEs, α, can be calculated as:

$$\alpha = \frac{Oup}{\beta} = \frac{Inp}{K \times K}. \tag{7}$$

D. Fully-Pipelined Inter-Layer Streaming Mapping

With the hardware-guided progressive pruning, we are able to put all layers of the CNN model on the FPGA chip, and adopt the fully-pipelined inter-layer streaming accelerator, as shown in Fig. 1. All parameters can be kept in on-chip memory to avoid the overhead caused by off-chip memory access, and all layers are computed concurrently with low-bit precisions in a pipelined fashion: neighbor layers communicate data on-chip with FIFOs in a streaming fashion. All these lead to the improvement of the utilization in computational resources and the reduction in energy consumption. For each layer, we customize the hardware library components built in Section III-C to quickly build the hardware prototype. The individual layer customization feature not only allows us to configure the parallelism and bit-width precision flexibly for each layer, but also has higher adaptability for different layer types and layer sizes. Next, we present the latency and resource model for our CNN accelerator, and the parallelism setting of each layer through inter-layer matching, which is required in Algorithm 1.

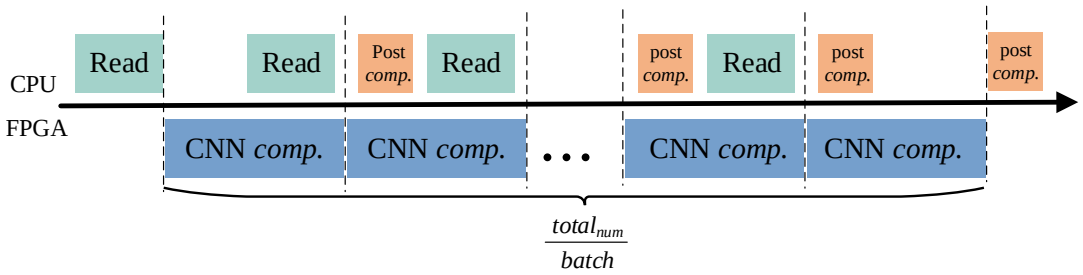


Fig. 6. Illustration of pipeline scheduling between sample reading, CNN computation, and post processing on the unified CPU-FPGA heterogeneous testing system. Each stage works on a batch of samples.

TABLE II. Detailed Description of SSDD Dataset

sensors	polarization	Resolution (m)	size of ships (num)			size of image (pixles)		images (num)	ships (num)
			small	medium	large	height	width		
RardarSat-2/TerraSAR-X/Sentinel-1	HH/VV/VH/HV	1-15	1529	936	76	190-526	214-618	1160	2546

1) CNN Accelerator Performance and Resource Modeling: Based on the proposed fully-pipelined inter-layer streaming architecture, the analysis model of performance and resource estimation for the whole CNN can be quickly constructed with the support of pre-modeled hardware library. Denote the i-th instantiated layer as L_i in a CNN model with D layers. Assume layer as L_i has θ_i components (e.g., SamePad, SWU, D-MVAU components), the latency of L_i can be computed as:

$$Lat_{L_i} = max\{Lat_{\sigma_p} | p = 1, ..., \theta_i\} \tag{8}$$

where Lat_{σ_p} represents the latency of the component p in L_i. For instance, the latency of the DWC layer is the maximum of three components: Samepad, SWU, and D-MVAU.

The resource cost of L_i is the sum of the resource consumed by all θ_i components:

$$Res_{L_i} = \sum_{p=1}^{\theta_i} Res_{\sigma_p} \tag{9}$$

Both Lat_{σ_p} and Res_{σ_p} are pre-modeled in our hardware library in Section III-C. They are highly related to the input parallelism Inp and output parallelism Oup, which we will explore next in Section III-D2.

Therefore, the overall CNN latency and resource cost can be obtained through the following two equations:

$$Lat_{CNN} = max\{Lat_{L_i} | i = 1, 2, ..., D\} \tag{10}$$

$$Res_{CNN} = \sum_{i=1}^{D} Res_{L_i} \tag{11}$$

2) Parallelism Tuning based on Inter-Layer Matching: Since almost all computational costs for CNNs are concentrated in the Matrix-Vector-Activation Unit (S-MVAU or D-MVAU), the configuration of Inp and Oup in each layer can determine the latency of the entire CNN accelerator. In this section, we propose the parallelism settings strategy based on inter-layer matching to meet the target latency. The guiding principles can be summarized as the following. First, the slowest layer's latency is equal to (or slightly lower than) the target latency. Second, the Inp of the next layer is always equal to the Oup of the previous layer, except for the first layer whose Inp is determined by its number of input channels. That is, as soon as the new result of the previous layer is generated, the next layer can be initiated immediately, thereby minimizing the initiation interval (II) to implement the inter-layer deep pipeline. Third, we allow some flexibility of the Oup of each layer, which can be explored among all the submultiples of the number of its output channels that satisfy the target latency with Eq. (2). Although some larger Oup values givers lower latency than the target latency and consumes more resources for the current layer, it gives subsequent layers more opportunities to adjust Inp and Oup to use less resources within the target latency. Finally, according to Eq. (11), the parallelism settings with the lowest resource cost are selected as the optimal configuration for the model.

E. Unified and Automated Heterogeneous Testing System

To quickly verify the on-board performance of our generated hardware accelerator, we construct a unified and automated CPU-FPGA testing system. It consists of reading input samples on the CPU (Read), computing the CNN on the FPGA (CNN *comp.*) and post-processing of non-CNN portion (post *comp.*) on the CPU. After the most time-consuming CNN computation is accelerated on the FPGA, the overhead of the remaining two stages becomes non-negligible. Therefore, we adopt a pipeline scheduling between these three stages, shown in Fig. 6. Assuming that the total number of input samples to be tested is $total\_num$ and $batch$ samples are processed each time. The optimization strategy is summarized as follows. First, in the Read stage, num_read number of CPU threads are used to read $batch$ samples. Second, in the CNN *comp.* stage, the FPGA accelerator processes the CNN computations for the batch. Third, in the post *comp.* stage, num_post number of CPU threads are applied to process the results produced by the CNN accelerator on the FPGA. While the CPU reads the next batch of images and completes the post-processing of the previous batch, the FPGA processes the CNN computation for the current batch. Therefore, the FPGA is always working at full capacity. The throughput of the system can be as close as possible to the throughput of the FPGA accelerator, through the adjustment of the num_read

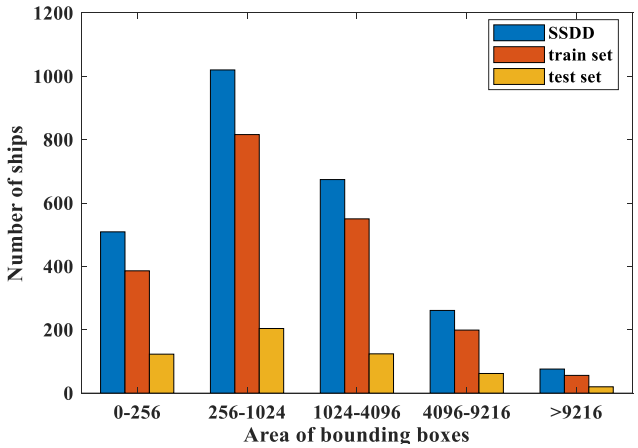


Fig. 7. Statistics of SSDD, training set and test set.

and num_post parameters.

Finally, we have developed Tcl scripts to instantiate the HLS-based FPGA accelerator with unified I/O interfaces and dynamic link library to start the CPU host code programmed in C language. This enables users to automatically realize heterogeneous deployment on the CPU and FPGA, while the underlying knowledge of the platform is not required.

Input	Operator	c	n	s
$416^2 \times 1$	SC 3×3	32	1	2
$208^2 \times 32$	DSC	64	1	1
$208^2 \times 64$	DSC	128	1	2
$104^2 \times 128$	DSC	128	1	1
$104^2 \times 128$	DSC	256	1	2
$52^2 \times 256$	DSC	256	1	1
$52^2 \times 256$	DSC	512	1	2(1)
$26^2 \times 512$ ($52^2 \times 512$)	DSC	512	5(2)	1
$26^2 \times 512$ ($52^2 \times 512$)	DSC	1024	1	2(1)
$13^2 \times 1024$ ($52^2 \times 1024$)	DSC	1024	1	1
$13^2 \times 1024$ ($52^2 \times 1024$)	SC 1×1	25	-	-

Fig. 8. The reference CNN model based on MobileNetV1. The highlighted part in blue and orange represents the details of two changes in the SAR-aware model adapting. The first column represents the IFM size for each Operator in the second column. Each row describes a sequence of n repeated identical layers. c denotes the number of output channels of each operator. The DWC of each operator has a stride, s.

IV. Experimental Results

A. Experimental Setup

1) Dataset: We use the SSDD [13] dataset constructed by Li *et al.*, which has been widely used for SAR ship detection. It followed the similar procedure of PASCAL VOC to collect SAR images by the RadarSat-2, TerraSAR-X, and Sentine-1 satellites. As listed out in Table II, it consists of 1,160 different single single-polarization images with 2,540 multiscale ships, ranging from 1m to 15m. Each image has an average of 2.03 ships, the smallest ship is about 7×7 pixels, while the largest is about 211×298 pixels. According to the area of the bounding boxes, the multiscale ships in SSDD is separated into 1,529 small ships (below 32 × 32 pixels), 935 medium ships (from 32 × 32 pixels to 96 × 96 pixels), and 76 large ships (above 96 × 96 pixels). This dataset focuses more on the detection capability of the network for the typical small-scale and medium-scale ships.

For the balance of the offshore and inshore ships, SSDD is divided into the training set (80%) and test set (20%) by applying the principles in [22]. Concretely, the images ending with 1 and 9 in the dataset are treated as test set and the rest of images are the training set. The statistical results of ship scales in SSDD, training set and test set are shown in Fig 7. In our experiments, data augmentation strategies, such as random cropping, flipping, translating and mirroring, are utilized to improve the robustness of the CNN training.

2) Reference CNN Models: The reference CNN models we start with are three lightweight models—MobileNetV1 [32], MobileNetV2 [31], and SqueezeNet [33]—which have state-of-the-art performance in object detection or image classification for optical images.

The network structures of the three models are shown in Fig. 8-10. MobilenetV1 adopted a DSC (depthwise separable convolution) composed of a 3×3 DWC (depth-wise convolution) and a 1×1 SC (standard convolution) to replace the standard 3×3 SC, significantly reducing the parameters and computations. Based on MobileNetV1, MobilenetV2 proposed a novel resource-efficient bottleneck block composed of 1×1 SC, 3×3 DWC, and linear 1×1 SC. For SqueezeNet, its well-designed core block is the Fire block composed of a squeeze 1×1 SC layer, followed by a expand layer including a 1×1 SC and a 3×3 SC. Note that we replace the additional bias of each convolutional layer with a BN layer, adopt RELU6 for all activation functions, and adjust the kernel size of the max-pooling layer from 3×3 to 2×2 in the SqueezeNet. These amendments increase the original SqueezeNet detection accuracy AP by 1.4% (from 85.2% to 86.6% in Table IV).

Our experiment utilizes the backbone of three models as feature extractors and employs the YOLOv2 back-end using five different anchor boxes for ship detection. These anchor boxes are obtained by applying K-means clustering. The backbone configuration of the three models is maintained except that the input images are replaced with SAR images that appear as grayscale images with small ship objects.

3) Input and Hyper Parameters in Algorithm 1: As shown in Table III, Algorithm 1 takes our adapted models with different convolutional layers ($D = 21$ for MobileNetV1, $D = 28$ for MobileNetV2, $D = 24$ for SqueezeNet) as the input network. We set the maximum accuracy loss $\epsilon = 3\%$, target latency $Lat_{tar} = 3.2ms$, $Lat_{tar} = 1.6ms$, or $Lat_{tar} = 0.7ms$, and the available DSP resource $Res_{total} = 3,600$ (DSP is the bottleneck resource in our experiments). Note that an appropriately large ϵ can avoid the early termination of the pruning process caused by the short-term negative accuracy fluctuation.

Input	Operator	t	c	n	s
$416^2 \times 1$	SC 3×3	-	32	1	2
$208^2 \times 32$	bottleneck	1	16	1	1
$208^2 \times 16$	bottleneck	6	24	2	2
$104^2 \times 24$	bottleneck	6	32	3	2
$52^2 \times 32$	bottleneck	6	64	4(2)	2(1)
$26^2 \times 64$ ($52^2 \times 64$)	bottleneck	6	96	3(1)	1
$26^2 \times 96$ ($52^2 \times 96$)	bottleneck	6	160	3	2(1)
$13^2 \times 160$ ($52^2 \times 160$)	bottleneck	6	320	1	1
$13^2 \times 320$ ($52^2 \times 320$) ($52^2 \times 96$)	SC 1×1	-	1280	1	1
$13^2 \times 1280$ ($52^2 \times 1280$)	SC 1×1	-	25	-	-

Fig. 9. The reference CNN model based on MobileNetV2. The highlighted part in blue and orange represents the details of two changes in the SAR-aware model adapting. The first column represents the IFM size for each Operator in the second column. Each row describes a sequence of n repeated identical layers. c denotes the number of output channels of all layers in the same sequence. t represents the expansion factor of the output channel compared with its input channel in the first 1×1 SC in each bottleneck block. The first layer of each sequence has a stride s and others use stride 1. The specific definitions of t, c, n and s have been described in detail in [31].

Input	Operator	$c1$	$c2$	s
$416^2 \times 1$	SC 3×3	-	96	2
$208^2 \times 96$	Maxpool	-	96	2
$104^2 \times 96$	Fire	16	64	1
$104^2 \times 128$	Fire	16	64	1
$104^2 \times 128$	Fire	32	128	1
$104^2 \times 256$ ($104^2 \times 128$)	Maxpool	-	256 (128)	2
$52^2 \times 256$ ($52^2 \times 128$)	Fire	32	128	1
$52^2 \times 256$	Fire	48	192	1
$52^2 \times 384$	Fire	48	192	1
$52^2 \times 384$	Fire	64	256	1
$52^2 \times 512$	Maxpool	-	512	2
$26^2 \times 512$ ($52^2 \times 512$)	Fire	64	256	1
$26^2 \times 512$ ($52^2 \times 512$)	SC 1×1	-	25	-

Fig. 10. The reference CNN model based on SqueezeNet. The highlighted part in blue and orange represents the details of two changes in the SAR-aware model adapting. The first column represents the IFM size for each Operator in the second column. $c1$ denotes the number of output channels of the squeeze convolution in each Fire Operator. $c2$ represents the the number of output channel of the operator–SC 1×1 and Maxpool, or the number of output channel of two convolutions with different kernel sizes in the expand layer of each Fire Operator. Each max-pooling has a stride s and a kernel size of 2.

For adapted MobileNetV1, we treat the DSC block including DWC 3×3 and SC 1×1 (e.g., layer 1-2 in Table V) as one pruning layer, which preserves the structural advantage of the existing model and accelerates the search process. Similarly, the bottleneck block in the adapted MobileNetV2—that is composed of SC 1×1, DWC 3×3, and linear SC 1×1 layers (e.g., layer 3-5 in Table VI)—is regarded as one pruning layer. The Fire block consisting of a squeeze SC layer and a expand layer (e.g., layer 2-4 in Table VII) is one pruning layer of adapted SqueezeNet. The AP (average precision) based on an IoU (Intersection over Union) threshold of 0.5 is used to quantitatively evaluate the detection accuracy.

Different pruning rates are set for three models in the coarse-grained global pruning. Considering the structural characteristic, the interpretation of the pruning rate is different for each model. For MobileNetV1, $rate = \frac{1}{4}/\frac{1}{2}$ means $\frac{1}{4}$ of output filters are removed for the first eight DSC pruning layers and $\frac{1}{2}$ of output filters for the last two DSC pruning layers. For MobileNetV2, $rate = \frac{1}{4}$ represents $\frac{1}{4}$ of output filters of each layer (not pruning layer) is pruned except the first two layers (L=1,2 in Table VI). For Squeezenet, $rate = \frac{1}{2}/\frac{1}{4}$ denotes the $\frac{1}{2}$ of output filters in the first convolutional layer is removed, and $\frac{1}{4}$ of the output filters of two convolution layers in the expand layer of each Fire pruning layer (e.g., layer 6 and layer 7 in Table VII) are pruned.

In the mixed-precision quantization, 4-bit signed weights (W) and 3-bit $\sim$ 6-bit activations (A) are employed, except that the input layer of the model uses 8-bit unsigned data and the output layer use 32 bit signed data.

TABLE III. The Detailed Settings of Input and Hyper Parameters Used in Algorithm 1. (U) and (S) denote unsigned and signed integer, respectively.

Input or Hyper Param.	MobilenetV1	MobilenetV2	SqueezeNet
D	21	28	24
Lat_{tar}	1.6ms/0.7ms	1.6ms/0.7ms	3.2ms/1.6ms
ϵ	3%		
Res_{total}	3600		
$iter$	200	100	100
$rate$	$\frac{1}{4}/\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{2}/\frac{1}{4}$
A	(U)3	detailed in Table V	(U)3
W	(S)4		
f_k	8		

In the fine-grained progressive pruning, the iteration number is set as $iter = 200$ for MobileNetV1, $iter = 100$ for MobileNetV2 and SqueezeNet. During each iteration, $f_k = 8$ is used. In MobileNetV1, f_k indicates the pruning number of output filters in the pruning DSC layer. In MobileNetV2, f_k output filters of DWC in the pruning layer are sequentially pruned without changing the input and output channels of the pruning bottleneck layer. In SqueezeNet, the f_k output filters of two convolution layer in the expand layer of each Fire pruning layer is pruned in each iteration.

We use Pytorch [56] to specify all models and training scripts on the Nvidia RTX 2080Ti GPU. And all training processes use Adam optimizer with appropriate learning rate and weight decay.

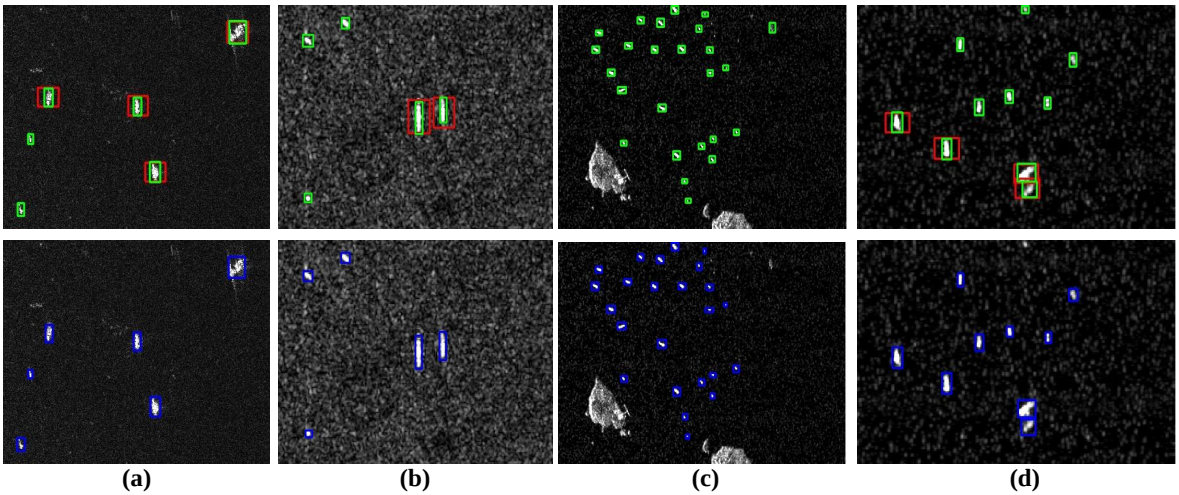


Fig. 11. The small-scale SAR ship detection results for offshore scenes. The first row shows the detection results of the initial MobileNetV2 model (highlighted by RED) and our modified model (highlighted by GREEN). The second row shows the ground truths, which are highlighted by BLUE.

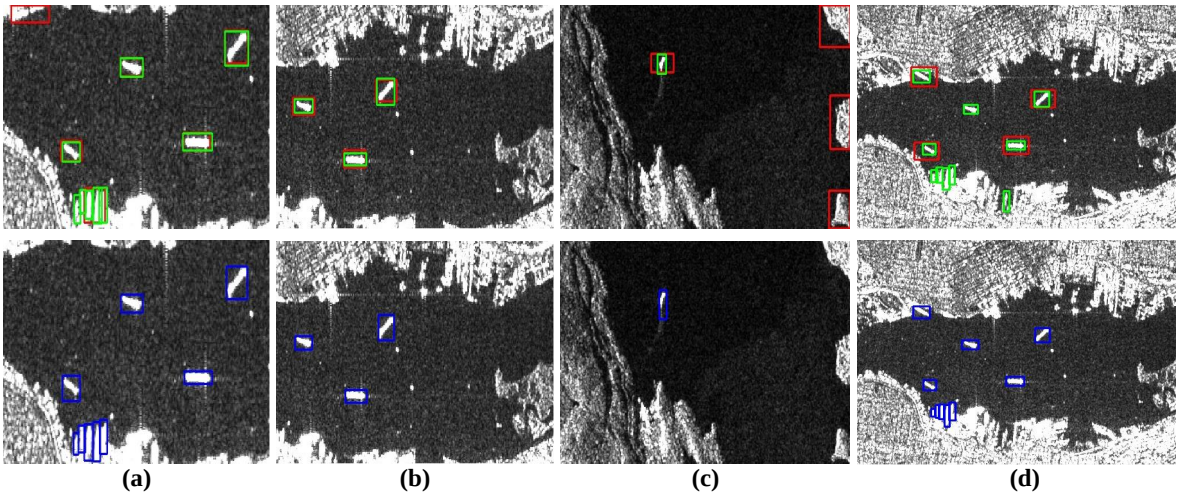


Fig. 12. The small-scale SAR ship detection results for inshore scenes. The first row shows the detection results of the initial MobileNetV2 model (highlighted by RED) and our modified model (highlighted by GREEN). The second row shows the ground truths, which are highlighted by BLUE.

4) Hardware Setup: The proposed hardware architecture is synthesized and implemented using Xilinx Vitis HLS and Vivado version 2020.2. We select the Xilinx VC709 evaluation board that has a Xilinx Virtex7 690T FPGA chip as the target platform to evaluate our accelerator. The board has 433,200 LUTs, 174,200 LUTRAMs, 866,400 FFs, 1,470 BRAMs and 3,600 DSPs. The host code runs on a 6-core Intel(R) Core(TM) i5-8400 (@2.80GHz) CPU server. For the comparison to the GPU, we use a Nvidia RTX 2080Ti GPU and use cuDNN library to implement the same pruned CNN models.

B. Results for SAR-Aware Model Adapting

Table IV lists the detection accuracy, parameters and model complexity of each stage in the model adapting. The detection accuracy of each original model is poor. For instance, the detection accuracy of the original reference MobileNetV2 is only 48.7%. Some test results of some typical small-scale ships in offshore scenes and inshore scenes are visualized in Fig. 11 and Fig. 12, respectively. We notice that the original MobileNetV2 has a lot of missed detections and false alarms. In addition, there is a certain degree of positioning error for the correctly detected ships.

TABLE IV. Performance Results on Detection Accuracy, Model Size and Computational Complexity at Different Stages. The value within round brackets in AP column refers to an improvement or loss in AP over the previous stage. Params refers to the total amount of parameters of convolutional layers and BN layers in the CNN model. Complexity refers to the number of multipy-accumulates of convolutional layers and BN layers in the CNN model.

Model	CNN	AP(%)	Param(MB)	Complexity(GOP)
MobilenetV1	reference model	50.7	12.93	1.95(FP32)
	modified model1	96.1(+45.4)	12.93	9.04(FP32)
	modified model2	96.0(-0.1)	9.7	6.87(FP32)
	coarse-grain pruning	96.0(-0)	3.77	2.75(FP32)
	mixed-precision quantization	96.1(+0.1)	0.51	2.75(low-bit int)
MobilenetV2	reference model	48.7	8.97	1.03(FP32)
	modified model1	95.4(+46.4)	8.97	6.25(FP32)
	modified model2	95.1(-0.3)	1.42	1.18(FP32)
	coarse-grain pruning	94.9(-0.2)	0.84	0.71(FP32)
	mixed-precision quantization	94.7(-0.2)	0.13	0.71(low-bit int)
SqueezeNet	reference model	86.6	2.95	2.17(FP32)
	modified model1	96.1(+9.5)	2.95	2.59(FP32)
	modified model2	95.8(-0.3)	2.75	2.09(FP32)
	coarse-grain pruning	95.7(-0.1)	2.06	1.55(FP32)
	mixed-precision quantization	95.5(-0.2)	0.27	1.55(low-bit int)

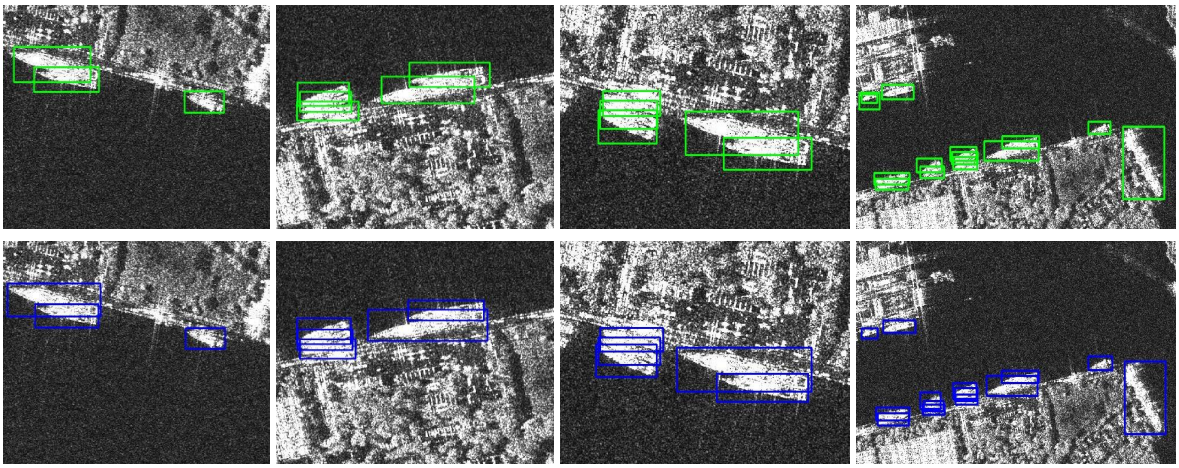


Fig. 13. The typical SAR ship detection results for complex inshore scenes for our modified model MobileNetV1 (highlighted by GREEN). The second row shows the ground truths, which are highlighted by BLUE.

As analyzed in Section III-A, this phenomenon is caused by the essential characteristics of densely clustered small objects but large detection scenes in SSDD. While the original network usually loses a lot of small target information due to the small output feature map. Fortunately, our experiments found that the output feature size of the feature extractor is positively correlated with the ability to detect small objects (Fig. 2 in Section III.A). That is, the larger the output feature size, the stronger the capability to detect small objects. Intuitively, the original SqueezeNet has relatively high detection accuracy (86.6% of AP) compared with the other two models because its backbone network has a larger output feature size. Based on this observation, in modified model 1, with changes highlighted by BLUE in Fig. 8-10, we modify the output feature size accordingly using values denoted in Fig. 2. As a result, its accuracy is improved to more than 95.4% as shown in Table IV. Such a sharp rise in accuracy is at the expense of higher computational complexity than the original reference model. For example, the complexity of the modified model 1 for MobileNetV1 is increased to 4.6 $\times$ that of the reference model.

Compared with the vast and complex ImageNet, SSDD is featured by a small dataset, a single category, and relatively few scenes without much feature information. Therefore, modified model 1 still has a lot of redundant information due to too deep network. Therefore, we further remove some repeated blocks, which are highlighted by ORANGE in Fig. 8-10. This results in only less than 0.3% accuracy loss, shown as modified model 2 in Table IV, while the parameters and complexity of MobileNteV2 are reduced by approximately 6.3$\times$ and 5.3$\times$.

To visually observe the performance improvement caused by SAR-aware adapting, we give some typical small-scale ship detection results of MobileNetV2. As shown in Fig. 11 and Fig. 12, even in complex scenes with strong speckle noise (Fig. 11 (b)), ocean background (Fig. 12 (c)), and land clutter (Fig. 12 (d)), our modified model 2 can detect and locate more small-scale ships, much better than the original MobileNetV2.

As listed in Table IV, more than 95% of the AP values for three modified model2 illustrate that our adapted models obtains good results on most test samples. For instance, Fig. 13 shows more test results of our adapted MobileNetV1 for different complex sea conditions and positions. It can be observed that all ship targets are correctly detected except for a false alarm (Fig. 13 (d)) and slight positioning error (Fig. 13 (b)). These results intuitively prove the strong robustness of our model in different environments.

TABLE V. Model Structure Obtained by Hardware-Guided Progressive Pruning for MobileNetV1. The first to sixth columns in the table describe the layer ID, the number of rows/columns of its IFM, the operation in the layer, the number of input channels of its IFM after coarse-grained pruning, two fine-grained pruning strategies respectively with $Lat_{tar} = 1.6ms$ and $Lat_{tar} = 0.7ms$.

L	*R/C*	operator	N_coarse	N_fine_1	N_fine_2
0	416	SC 3×3	1	1	1
1	208	DWC 3×3	32	**16**	**24**
2	208	SC 1×1	32	**16**	**24**
3	208	DWC 3×3	48	**24**	**8**
4	104	SC 1×1	48	**24**	**8**
5	104	DWC 3×3	96	**64**	**24**
6	104	SC 1×1	96	**64**	**24**
7	96	DWC 3×3	96	96	96
8	96	SC 1×1	96	96	96
9	52	DWC 3×3	192	192	**112**
10	52	SC 1×1	192	192	**112**
11	52	DWC 3×3	192	**96**	**168**
12	52	SC 1×1	192	**96**	**168**
13	52	DWC 3×3	384	**224**	**120**
14	52	SC 1×1	384	**224**	**120**
15	52	DWC 3×3	384	**96**	**120**
16	52	SC 1×1	384	**96**	**120**
17	52	DWC 3×3	384	**256**	**168**
18	52	SC 1×1	384	**256**	**168**
19	52	DWC 3×3	512	**48**	**56**
20	52	SC 1×1	512	**48**	**56**
21	52	SC 1×1	512	512	512
-	52	Bounding Box	25	25	25

C. Results for Hardware-Guided Progressive Pruning

1) Results for Coarse-Grained Pruning: According to the pruning principle in Section III-B and the detailed pruning rates in Section IV-A, we performed coarse-grained global pruning on the three modified network backbones. The number of remaining input or output channels for each layer after the

TABLE VI. Model Structure Obtained by Hardware-Guided Progressive Pruning for MobileNetV2. The first to eighth columns in the table describe the layer ID, the number of rows/columns of its IFM, the operation in the layer, the number of input channels of its IFM after coarse-grained pruning, two fine-grained pruning strategies respectively with $Lat_{tar} = 1.6ms$ and $Lat_{tar} = 0.7ms$, and the number of input activation bits and output activation bits for each layer. (U) and (S) denote unsigned and signed integer, respectively.

L	*R/C*	Operator	N_coarse	N_fine_1	N_fine_2	A_i	A_o
0	416	SC 3×3	1	1	1	(U)8	(U)3
1	208	DWC 3×3	32	32	32	(U)3	(U)3
2	208	SC 1×1	32	32	32	(U)3	(S)4
3	208	SC 1×1	12	12	12	(S)4	(U)3
4	208	DWC 3×3	72	**32**	**48**	(U)3	(U)3
5	104	SC 1×1	72	**32**	**48**	(U)3	(S)4
6	104	SC 1×1	18	18	18	(S)4	(U)3
7	104	DWC 3×3	108	**60**	**36**	(U)3	(U)3
8	104	SC 1×1	108	**60**	**36**	(U)3	(S)4
-			Add to input of layer 6				
9	104	SC 1×1	18	18	18	(S)5	(U)3
10	104	DWC 3×3	108	84	84	(U)3	(U)3
11	52	SC 1×1	108	84	84	(U)3	(S)4
12	52	SC 1×1	24	24	24	(S)4	(U)3
13	52	DWC 3×3	144	144	144	(U)3	(U)3
14	52	SC 1×1	144	144	144	(U)3	(S)4
-			Add to input of layer 12				
15	52	SC 1×1	24	24	24	(S)5	(U)3
16	52	DWC 3×3	144	144	144	(U)3	(U)3
17	52	SC 1×1	144	144	144	(U)3	(S)4
-			Add to input of layer 15				
18	52	SC 1×1	24	24	24	(S)6	(U)3
19	52	DWC 3×3	144	144	144	(U)3	(U)3
20	52	SC 1×1	144	144	144	(U)3	(S)4
21	52	SC 1×1	48	48	48	(S)4	(U)3
22	52	DWC 3×3	288	**168**	**200**	(U)3	(U)3
23	52	SC 1×1	288	**168**	**200**	(U)3	(S)4
-			Add to input of layer 21				
24	52	SC 1×1	48	48	48	(S)5	(U)3
25	52	DWC 3×3	288	**128**	**120**	(U)3	(U)3
26	52	SC 1×1	288	**128**	**120**	(U)3	(S)4
27	52	SC 1×1	72	72	72	(S)4	(U)3
28	52	SC 1×1	960	960	960	(S)3	(S)32
-	52	Bounding Box	25	25	25	-	-

TABLE VII. Structure Obtained by Hardware-Guided Progressive Pruning for SqueezeNet. The first to sixth columns in the table describe the layer ID, the number of rows/columns of its OFM, the operation in the layer, the number of output channels of its OFM after coarse-grained pruning, two fine-grained pruning strategies respectively with $Lat_{tar} = 3.2ms$ and $Lat_{tar} = 1.6ms$.

L	*OR/OC*	operator	M_coarse	M_fine_1	M_fine_2
0	208	SC 3×3	48	48	48
1	104	Maxpool	48	48	48
2	104	SC 1×1	16	16	16
3	104	SC 1×1	48	48	48
4	104	SC 3×3	48	48	48
-		concat[layer3, layer4]			
5	104	SC 1×1	16	16	16
6	104	SC 1×1	48	**40**	**24**
7	104	SC 3×3	48	**40**	**24**
-		concat[layer6, layer7]			
8	52	Maxpool	96	**80**	**48**
9	52	SC 1×1	32	32	32
10	52	SC 1×1	96	96	96
11	52	SC 3×3	96	96	96
-		concat[layer10, layer11]			
12	52	SC 1×1	48	48	48
13	52	SC 1×1	144	**88**	**24**
14	52	SC 3×3	144	**88**	**24**
-		concat[layer13, layer14]			
15	52	SC 1×1	48	48	48
16	52	SC 1×1	144	**48**	**40**
17	52	SC 3×3	144	**48**	**40**
-		concat[layer16, layer17]			
18	52	SC 1×1	64	64	64
19	52	SC 1×1	192	**80**	**72**
20	52	SC 33	192	**80**	**72**
-		concat[layer19, layer20]			
21	52	SC 1×1	64	64	64
22	52	SC 1×1	192	**56**	**32**
23	52	SC 3×3	192	**56**	**32**
-		concat[layer22, layer23]			
24	52	SC 1×1	25	25	25

coarse-grained global pruning is shown in the fourth column of Table V-VII. As presented in Table IV, the AP decreases by an additional of 0%, 0.2% and 0.1%, while parameter/complexity are reduced by about 61%/60%, 41%/40% and 25%/26% respectively.

2) Results for Mixed-Precision Quantization: In the mixed-precision quantization, 4-bit signed weights (W) and 3-bit $\sim$ 6-bit activations (A) are employed, except that the input layer of the model uses unsigned 8-bit data and the output layer (L=21 in Table V, L=28 in Table VI and L=24 in Table VII) uses 32-bit. Specifically, all activation bit-width using RELU6 in MobilenetV1 and SqueezeNet is set to unsigned 3-bit. For MobileNetV2, the detailed number of bits for the activations in each layer is summarized in the seventh and eighth column of Table VI. For instance, we set the output activation bit-width obtained by SC (e.g., L=3, 6, 9 in Table VI) and DWC (e.g., L=1, 4, 7), as well as following Relu6 unit to unsigned 3-bit. And we set the output activation obtained by linear SC (e.g., L=5, 11, 20) to signed 4-bit. In addition, higher activation bit-widths are applied when the addition operation is performed by shortcut. As shown in Table IV, the mixed-precision quantization incurs an additional less than 0.2% accuracy loss, but further greatly reduces the parameter size.

3) Results for Fine-Grained Pruning: In the fine-grained progressive pruning, the maximum accuracy loss ϵ is set to 3%. Fig. 14 presents the trends of accuracy and resource cost (number of DSPs) for three models with different target latency. Intuitively, the reduction of resource cost is at the expense of decreasing detection accuracy. However, we also observe the fluctuation of detection accuracy in the short-term training, which could give a better trade-off between the accuracy and resource cost. As highlighted in Fig. 14, for the corresponding target latency settings, we pick out six candidates (CNN1@1.6 and CNN2@0.7 for MobileNetV1, CNN3@1.6 and CNN4@0.7 for MobileNetV2, CNN5@3.2 and CNN6@1.6 for SqueezeNet) respectively, with similar detection accuracy, parameter size and complexity. The detailed network structure after the pruning process is summarized in the fifth and sixth columns of Table V-VII.

D. Results for Hardware Performance

The final board-level deployment results are listed in Table VIII. As shown in the third column in Table, it can be observed

TABLE VIII. Performance Evaluation of Board-Level Deployment within Our Proposed End-to-End Framework OSCAR-RT.

Model	Lat_{Tar} (ms)	AP (%)	Frequency (MHZ)	Batch	GOP	Peak FPS	GOPS	System FPS	Resource Utilization					
									LUT	LUTRAM	BRAM	FF	DSP	Predict_DSP
MobilenetV1	CNN1@1.6	94	250	25	0.57	388	221	385	123173 (28.43%)	5954 (3.42%)	263 (17.89%)	222453 (25.68%)	1009 (28.03%)	954 (26.5%)
	CNN2@0.7	94	250	25	0.45	663	298	652	154126 (35.58%)	7199 (4.13%)	313.50 (21.33%)	273634 (31.58%)	2311 (64.19%)	2183 (60.64%)
MobileNetV2	CNN3@1.6	93.3	250	50	0.56	383	214	380	137094 (31.65%)	6177 (3.55%)	247.50 (16.84%)	239613 (27.66%)	1023 (28.42%)	948 (26.33%)
	CNN4@0.7	93.3	250	50	0.56	653	366	636	196924 (45.46%)	7906 (4.54%)	319.5 (21.73%)	320828 (37.03%)	2496 (69.33%)	2378 (66.06%)
SqueezeNet	CNN5@3.2	94.6	250	25	0.8	244	195	237	132958 (30.69%)	7567 (4.34%)	311 (21.16%)	239103 (27.60%)	1986 (55.17%)	1897 (52.69%)
	CNN6@1.6	92.8	250	25	0.6	391	235	384	160387 (37.02%)	8138 (4.67%)	264.50 (17.99%)	292672 (33.78%)	2736 (76%)	2707 (75.19%)

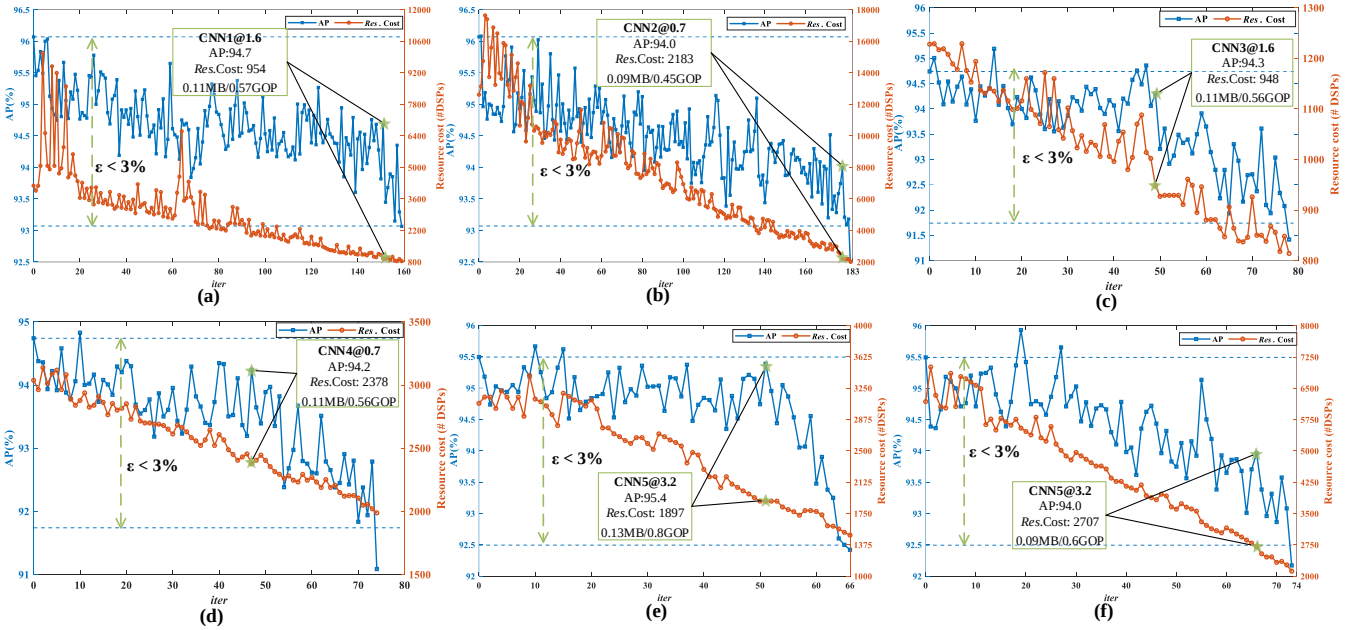


Fig. 14. Accuracy and resource cost of candidate CNNs found in the progressive fine-grained pruning for MobileNetV1 with Lat_{tar}=1.6 ms in (a) and Lat_{tar}=0.7 ms in (b), MobileNetV2 with Lat_{tar}=1.6 ms in (c) and Lat_{tar}=0.7 ms in (d), and SqueezeNet with Lat_{tar}=3.2 ms in (e) and Lat_{tar}=1.6 ms in (f). $iter$=0 refers to the model obtained after coarse-grained pruning and mixed-precision quantization.

that the AP values in the test set bring an additional accuracy loss compared with their corresponding CNN algorithm accuracy, which is caused by replacing the division operation with the right shifter in the hardware implementation of the ACTU unit. The peak FPS (frames per second) measures the performance of the CNN inference on the FPGA and the system FPS is obtained including reading the input, computing the CNN inference and post-processing the detection results, by using 1K 416×416 SAR images with a batch size of 50 or 25. At a clock frequency of 250MHz, CNN5@3.2 based on the SqueezeNet achieves the highest AP, reaching 94.6% with 237 system FPS and 4.4W power, while CNN2@0.7 based on the MobileNetV1 achieves the highest system FPS at 652 FPS with 94% AP and 5.8W. CNN4@0.7 based on the MobileNetV2 has the highest GOPS at 366 GOPS with 93.3% AP and 6.6W. In addition, the DSP usage is the bottleneck of the hardware implementation and the estimated DSP is very close to the real DSP utilization, indicating the effectiveness of our proposed accelerator modeling.

1) Comparison to GPU Implementation: To the best of our knowledge, the existing CNN based on SAR ship detection work [18]–[22], [51] only study the GPU solutions with the detection accuracy as the optimization objective , which is not suitable for on-satellite processing due to the power and radiation considerations. Meanwhile, we also compare the CNN inference latency of our proposed FPGA accelerator design to that running on a Nvidia RTX 2080Ti GPU using a batch size of 1. For the GPU implementation, we use the same pruned MobileNetV1,MobileNetV2, and SqueezeNet as that used in our FPGA acceleration, and use Nvidia cuDNN library to implement it. The dynamic power refers to the difference between running and not running CNN inference on the FPGA (or GPU). Fig. 15 shows the aggressive advantages of our FPGA implementation in latency and power consumption compared with GPU implementation. For instance, compared to the GPU implementation, our FPGA accelerator in MobileNetV2 achieves 2.4× and 4.2× lower latency, and consumes 11.6× and 7.9× less power, for CNN3@1.6 and

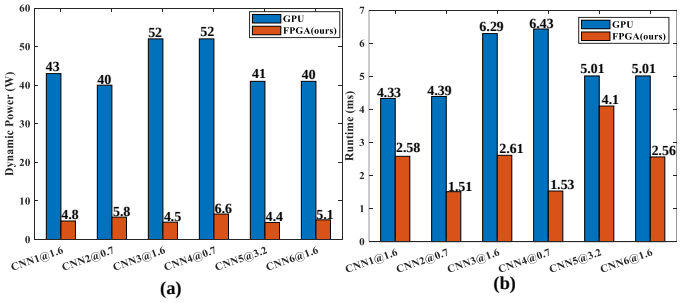


Fig. 15. Performance comparisons of different platforms. The dynamic power in (a) refers to the CNN difference between running and not running CNN inference on the FPGA or GPU. The runtime is measured by detecting one 416×416 SAR image in (b).

CNN4@0.7 models, respectively. It is worth mentioning that due to the portable and configurable hardware architecture design, our implementation is easy to assemble into one radiation-resistant chip suitable for satellite environment.

V. Conclusion and Future Work

In this work, we have proposed OSCAR-RT, the first end-to-end algorithm/hardware co-design framework for highly-accurate and real-time on-satellite CNN-based SAR ship detection. Inside OSCAR-RT, we have proposed the SAR-aware CNN model adapting, hardware-guided progressive pruning, portable and reusable hardware library, and fully-pipelined inter-layer streaming accelerator mapping on FPGAs. Experimental results using the several widely adopted networks and SSDD SAR ship detection dataset have demonstrated that OSCAR-RT can simultaneously produce an accurate and hardware-friendly CNN model and an ultra-efficient FPGA-based hardware accelerator. On the Xilinx VC709 FPGA evaluation board, OSCAR-RT using our adapted MobileNetV1 model achieves an average precision of 94%, a detection speed of 652 frames per second, while consuming about 5.8W low power. In addition, compared to the GPU implementation on a Nvidia RTX 2080Ti GPU, it achieves 2.9× lower latency and consumes more than 6.8× less dynamic power.

In future work, we plan to adapt our framework to support more CNN models and more remote sensing applications such as instance segmentation. Moreover, we plan to fully automate the co-design process without manual intervention for those hyper parameters and open source our framework.

References

[1] G. Gao, K. Ouyang, Y. Luo, S. Liang, and S. Zhou, "Scheme of parameter estimation for generalized gamma distribution and its application to ship detection in sar images," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 3, pp. 1812–1832, 2016.

[2] M. Yang and C. Guo, "Ship detection in sar images based on lognormal β-metric," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 9, pp. 1372–1376, 2018.

[3] X. Leng, K. Ji, S. Zhou, and H. Zou, "Noncircularity parameters and their potential in ship detection from high resolution sar imagery," in *IEEE Inter. Geosci. Remote Sens. Symp.(IGARSS)*. IEEE, 2017, pp. 1876–1879.

[4] A. C. Copeland, G. Ravichandran, and M. M. Trivedi, "Localized radon transform-based detection of ship wakes in sar images," *IEEE Trans. Geosci. Remote Sens.*, vol. 33, no. 1, pp. 35–45, 1995.

[5] F. C. Robey, D. R. Fuhrmann, E. J. Kelly, and R. Nitzberg, "A cfar adaptive matched filter detector," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 28, no. 1, pp. 208–216, 1992.

[6] W. Zhou, J. Xie, G. Li, and Y. Du, "Robust cfar detector with weighted amplitude iteration in nonhomogeneous sea clutter," *IEEE Trans. Aerosp. Electron. Sys.*, vol. 53, no. 3, pp. 1520–1535, 2017.

[7] C. P. Schwegmann, W. Kleynhans, and B. P. Salmon, "Manifold adaptation for constant false alarm rate ship detection in south african oceans," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 7, pp. 3329–3337, 2015.

[8] S. Tian, C. Wang, and H. Zhang, "A segmentation based global iterative censoring scheme for ship detection in synthetic aperture radar image. doc," in *Proc. Int. Geosci. Remote Sens. Sympos.* IEEE, 2016, pp. 6513–6516.

[9] W. Zhou, J. Xie, K. Xi, and Y. Du, "Modified cell averaging cfar detector based on grubbs criterion in non-homogeneous background," *IET Radar Sonar Navigation*, vol. 13, no. 1, pp. 104–112, 2018.

[10] J. Ai, Q. Luo, X. Yang, Z. Yin, and H. Xu, "Outliers-robust cfar detector of gaussian clutter based on the truncated-maximum-likelihood-estimator in sar imagery," *IEEE Trans. Intell. Transp. Sys.*, vol. 21, no. 5, pp. 2039–2049, 2019.

[11] J. Ai, X. Yang, J. Song, Z. Dong, L. Jia, and F. Zhou, "An adaptively truncated clutter-statistics-based two-parameter cfar detector in sar imagery," *IEEE J. Ocean. Eng.*, vol. 43, no. 1, pp. 267–279, 2017.

[12] J. Ai, Y. Mao, Q. Luo, M. Xing, K. Jiang, L. Jia, and X. Yang, "Robust cfar ship detector based on bilateral-trimmed-statistics of complex ocean scenes in sar imagery: A closed-form solution," *IEEE Trans. Aerosp. Electron. Sys.*, vol. 57, no. 3, pp. 1872–1890, 2021.

[13] J. Li, C. Qu, and J. Shao, "Ship detection in sar images based on an improved faster r-cnn," in *2017 SAR in Big Data Era: Models, Methods and Applications (BIGSARDATA)*, 2017, pp. 1–6.

[14] S. Wei, X. Zeng, Q. Qu, M. Wang, H. Su, and J. Shi, "Hrsid: A high-resolution sar images dataset for ship detection and instance segmentation," *IEEE Access*, vol. 8, pp. 120 234–120 254, 2020.

[15] 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, 2019.

[16] 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, 2020.

[17] J. Jiao, Y. Zhang, H. Sun, X. Yang, X. Gao, W. Hong, K. Fu, and X. Sun, "A densely connected end-to-end neural network for multiscale and multiscene sar ship detection," *IEEE Access*, vol. 6, pp. 20 881–20 892, 2018.

[18] Y. Wang, C. Wang, H. Zhang, Y. Dong, and S. Wei, "Automatic ship detection based on retinanet using multi-resolution gaofen-3 imagery," *Remote Sens.*, vol. 11, no. 5, p. 531, 2019.

[19] Y.-L. Chang, A. Anagaw, L. Chang, Y. C. Wang, C.-Y. Hsiao, and W.-H. Lee, "Ship detection based on yolov2 for sar imagery," *Remote Sens.*, vol. 11, no. 7, 2019.

[20] 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.

[21] T. Zhang and X. Zhang, "High-speed ship detection in sar images based on a grid convolutional neural network," *Remote Sens.*, vol. 11, no. 10, 2019.

[22] Y. Chen, T. Duan, C. Wang, Y. Zhang, and M. Huang, "End-to-end ship detection in sar images for complex scenes based on deep cnns," *J Sensors*, vol. 2021, 2021.

[23] Z. Wu, B. Hou, B. Ren, Z. Ren, S. Wang, and L. Jiao, "A deep detection network based on interaction of instance segmentation and object detection for sar images," *Remote Sens.*, vol. 13, no. 13, 2021.

[24] J. Lei, G. Yang, W. Xie, Y. Li, and X. Jia, "A low-complexity hyperspectral anomaly detection algorithm and its fpga implementation," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 907–921, 2021.

[25] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, 2017, pp. 65–74.

[26] M. Blott, T. B. Preußer, N. J. Fraser, G. Gambardella, K. Obrien, Y. Umuroglu, M. Leeser, and K. Vissers, "Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 11, no. 3, pp. 1–23, 2018.

[27] M. A. Carreira-Perpinán and Y. Idelbayev, "learning-compression algorithms for neural net pruning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 8532–8541.

[28] M. Lin, L. Cao, S. Li, Q. Ye, Y. Tian, J. Liu, Q. Tian, and R. Ji, "Filter sketch for network pruning," *IEEE Tran. Neural Netw. Learn. Syst.*, 2021.

[29] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, and L. Shao, "Hrank: Filter pruning using high-rank feature map," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1529–1538.

[30] L. Lai, N. Suda, and V. Chandra, "Not all ops are created equal!" *arXiv preprint arXiv:1801.04326*, 2018.

[31] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf.Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.

[32] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[33] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[34] G. Goldstein, "False alarm regulation in log-normal and weibull clutter," *IEEE Trans. Aerosp. & Electron. Syst.*, vol. 28, no. 1, pp. 138–152, 1992.

[35] X. Qin, S. Zhou, H. Zou, and G. Gao, "A cfar detection algorithm for generalized gamma distributed background in high-resolution sar images," *IEEE Geosci. Remote Sens. Lett.*, vol. 10, no. 4, pp. 806–810, 2012.

[36] X. Zhou, R. Peng, and C. Wang, "A two-component k–lognormal mixture model and its parameter estimation method," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 5, pp. 2640–2651, 2014.

[37] X. Leng, K. Ji, K. Yang, and H. Zou, "A bilateral cfar algorithm for ship detection in sar images," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 7, pp. 1536–1540, 2015.

[38] S. Wiehle, D. Gnzel, and B. Tings, "Sar satellite on-board ship, wind, and sea state detection," in *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, 2021, pp. 8289–8292.

[39] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Adv. Neural Inf. Process. Syst.*, vol. 28, pp. 91–99, 2015.

[40] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE CVPR*, July 2017.

[41] Y. Zhang, Z. Xia, T. Zhang, Z. Zhao, D. Liu, H. Shi, and F. Yue, "Real-time processing of ship detection with spaceborne sar image based on multiple fpga," *IET Conference Proceedings*, pp. 1060–1065(5), January 2021.

[42] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, 2015, pp. 161–170.

[43] C. Zhang, G. Sun, Z. Fang, P. Zhou, P. Pan, and J. Cong, "Caffeine: Toward uniformed representation and acceleration for deep convolutional neural networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 11, pp. 2072–2085, 2019.

[44] J. Cong and J. Wang, "Polysa: Polyhedral-based systolic array auto-compilation," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*. IEEE, 2018, pp. 1–8.

[45] A. Sohrabizadeh, J. Wang, and J. Cong, "End-to-end optimization of deep learning applications," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, 2020, pp. 133–139.

[46] H. Ye, X. Zhang, Z. Huang, G. Chen, and D. Chen, "Hybriddnn: A framework for high-performance hybrid dnn accelerator design and implementation," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*. IEEE, 2020, pp. 1–6.

[47] X. Zhang, J. Wang, C. Zhu, Y. Lin, J. Xiong, W.-m. Hwu, and D. Chen, "Dnnbuilder: an automated tool for building high-performance dnn hardware accelerators for fpgas," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2018, pp. 1–8.

[48] X. Zhang, H. Ye, J. Wang, Y. Lin, J. Xiong, W.-m. Hwu, and D. Chen, "Dnnexplorer: a framework for modeling and exploring a novel paradigm of fpga-based dnn accelerator," in *Proc. 39th Int. Conf. Comput.-Aided Design (ICCAD)*, 2020, pp. 1–9.

[49] S. Williams, A. Waterman, and D. Patterson, "Roofline: an insightful visual performance model for multicore architectures," *Commum. ACM*, vol. 52, no. 4, pp. 65–76, 2009.

[50] K. Zhan, J. Guo, B. Song, W. Zhang, and Z. Bao, "DACSDC'20 1st place winner in fpga track," 2020.

[51] Z. Deng, H. Sun, S. Zhou, and J. Zhao, "Learning deep ship detector in sar images from scratch," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 6, pp. 4021–4039, 2019.

[52] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[54] SystemsETHZ, "Dacsdc'19 2nd place winner in fpga track," 2019. [Online]. Available: https://github.com/fpgasystems/spooNN

[55] X. Zhang, H. Lu, C. Hao, J. Li, B. Cheng, Y. Li, K. Rupnow, J. Xiong, T. Huang, H. Shi *et al.*, "Skynet: a hardware-efficient method for object detection and tracking on embedded systems," *Proc. Mach. Learn. Syst.*, pp. 216–229, 2020.

[56] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.

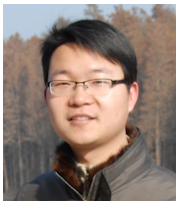
Geng Yang received the B.E. degree in Telecommunications Engineering from Xidian University, Xi'an, China in 2019. She is currently pursuing the Ph.D. degree in the Image Coding and Processing Center at State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, China.

Her research interests include remote sensing image processing, computer vision, efficient deep learning.

Jie Lei (M'19) received the M.S. degree in telecommunication and information systems and the Ph.D. degree in signal and information processing from Xidian University, China, in 2006 and 2010, respectively. He has been a Visiting Scholar at the Department of Computer Science of University of California, Los Angeles, USA, from 2014 to 2015. Currently, he is an Associate Professor at the school of Telecommunications Engineering, Xidian University, and is a member of the image coding and processing center at the State Key Laboratory of Integrated Services Networks, and is also with the Science and Technology on Electro-Optic Control Laboratory.

His research interests focus on image and video processing, computer vision, and customized computing for big-data applications.

Weiying Xie (M'18) received the B.S. degree in electronic information science and technology from University of Jinan in 2011. She received the M.S. degree in communication and information systems, Lanzhou University in 2014 and the Ph.D. degree in communication and information systems of Xidian University in 2017. Currently, she is an Associate Professor with the State Key Laboratory of Integrated Services Networks, Xidian University. She has published more than 30 papers in refereed journals. Her research interests include neural networks, machine learning, hyperspectral image processing, and high-performance computing.



Zhenman Fang received his PhD degree in Computer Science from Fudan University, China in 2014. Zhenman did his postdoc at UCLA from 2014 to 2017, and worked as a Staff Software Engineer at Xilinx, San Jose, from 2017 to 2019. Currently, he is an Assistant Professor in School of Engineering Science, Simon Fraser University, Canada. His recent research focuses on customizable computing with specialized hardware acceleration, including emerging application characterization and acceleration, novel accelerator-rich and near-data computing architecture designs, and corresponding programming, runtime, and tool support. He is a member of the ACM and IEEE.

Yunsong Li received the M.S. degree in telecommunication and information systems and the Ph.D. degree in signal and information processing from Xidian University, China, in 1999 and 2002, respectively. He joins the school of tetecommunications Engineering, Xidian Uniersity in 1999 where he is currently a Professor. Prof. Li is the director of the image coding and processing center at the State Key Laboratory of Integrated Services Networks.

His research interests focus on image and video processing and high-performance computing.

Xiaxuan Wang received the B.E. degree in Telecommunications Engineering from Xidian University, Xi'an, China in 2020. He is currently pursuing the M.S. degree in the Image Coding and Processing Center at State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, China.

His research interests include remote sensing image processing, target detection.

Xin Zhang received the B.E. degree in Telecommunications Engineering from Xidian University, Xi'an, China in 2019. She is currently pursuing the Ph.D. degree in the Image Coding and Processing Center at State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, China.

Her research interests include remote sensing image processing, target detection, and network pruning.