Article

A Real-Time FPGA Accelerator Based on Winograd Algorithm for Underwater Object Detection

Liangwei Cai [1], Ceng Wang [1] and Yuan Xu [2,*]

[1] College of Electronic and Information Engineering, Shenzhen University, Shenzhen 518000, China; cailw@szu.edu.cn (L.C.); cengwong@163.com (C.W.)
[2] College of Big Data and Internet, Shenzhen Technology University, Shenzhen 518000, China
[*] Correspondence: xuyuann@sztu.edu.cn; Tel.: +86-18938686682

Abstract: Real-time object detection is a challenging but crucial task for autonomous underwater vehicles because of the complex underwater imaging environment. Resulted by suspended particles scattering and wavelength-dependent light attenuation, underwater images are always hazy and color-distorted. To overcome the difficulties caused by these problems to underwater object detection, an end-to-end CNN network combined U-Net and MobileNetV3-SSDLite is proposed. Furthermore, the FPGA implementation of various convolution in the proposed network is optimized based on the Winograd algorithm. An efficient upsampling engine is presented, and the FPGA implementation of squeeze-and-excitation module in MobileNetV3 is optimized. The accelerator is implemented on a Zynq XC7Z045 device running at 150 MHz and achieves 23.68 frames per second (fps) and 33.14 fps when using MobileNetV3-Large and MobileNetV3-Small as the feature extractor. Compared to CPU, our accelerator achieves 7.5×–8.7× speedup and 52×–60× energy efficiency.

Keywords: underwater object detection; U-Net; MobileNetV3; Winograd algorithm

check for updates

Citation: Cai, L.; Wang, C.; Xu, Y. A Real-Time FPGA Accelerator Based on Winograd Algorithm for Underwater Object Detection. *Electronics* **2021**, *10*, 2889. https://doi.org/10.3390/electronics10232889

Academic Editor: Luis Parrilla

Received: 13 October 2021
Accepted: 18 November 2021
Published: 23 November 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Precise real-time object detection is crucially important in autonomous underwater vehicles (AUVs), which are wildly used for resource exploration, underwater rescue, and salvage. At present, sonar [1] and lidar [2] are the major approaches used in underwater object detection. However, both of these technologies have obvious shortcomings. For example, sonar suffers from low imaging resolution and poor short-range detection ability, while lidar can only detect the contour information of objects and has difficulties in distinguishing objects with a similar shape. With the rapid development of image processing, vision-based underwater object detection has gradually been a research hot-spot.

One of the primary challenges in vision-based underwater object detection is how to obtain underwater images with better quality. Because of the color cast and contrast degradation due to the absorption and scattering of light in water, it is difficult to perform high-accuracy underwater object detection for raw underwater images. Therefore, improving the quality of underwater images has become the first problem to be solved in underwater object detection. So far, many image enhancement algorithms have been applied into color restoration and edge enhancement of underwater images, such as fusion algorithm [3] and dark channel prior algorithm (DCP) [4]. In recent years, the encoder-decoder architecture convolutional neural network (CNN), U-Net, for instance, has achieved better performance in underwater images enhancement due to its lightweight structure and the powerful nonlinear approximation of CNN [5–7]. CNN is also much more popular than traditional methods in the field of object detection. Various high-accuracy CNN for object detection have been proposed over past few years. Compared to the standard CNN models, including YOLOv2 [8], Faster R-CNN [9] and GoogLeNet [10], the CNN based on depth-wise separable convolution, such as MobileNets [11–13], Xception [14], and ResNet [15], are more efficient by significantly reducing computations and parameters with limited loss in mean average precision (mAP).

An end-to-end underwater object detection CNN network combined U-Net and MobileNetV3-SSDLite is proposed in this paper. U-Net is used to restore underwater images to ensure the stable restoration quality under different imaging environment. MobileNetV3-SSDLite is used as object detection network to reduce the dependence on the computing ability of the deployment platform. In practical applications, field-programmable gate array (FPGA) is a popular option for designing AUVs due the conveniences of expanding peripheral interfaces and customizing special hardware control logic. Compared with focusing on balancing the computation parallelism and the memory bandwidth [16–24], the research focusing on optimizing the implementation of convolution computation onto FPGA have attracted more attention recently. Converted convolution into general matrix multiplication (GEMM) could reduce the times of memory access [25]. The fast Fourier transform (FFT) algorithm reduces the computational complexity of convolution, especially for the convolution with large filter kernels [26]. Based on the Chinese remainder theorem, the Winograd algorithm significantly reduces the multiplications required by 2-D convolution with small filter kernels [27]. Various convolution accelerators based on the Winograd algorithm have been proposed [28–31]. Moreover, [32,33] proposed a method of optimizing 2-stride convolution with the Winograd algorithm, which reduces the design complexity of the Winograd-based CNN accelerator and significantly enhances the computation efficiency. However, the Winograd algorithm is not the best optimization scheme for the upsampling in U-Net because of the sparse feature map after expansion. The processing strategy of the squeeze-and-excitation (SE) module in MobileNetV3 on Winograd-based convolution accelerator output results also affects the object detection speed of the proposed network significantly.

The main contributions of this article are as follows:

- An end-to-end CNN network combined U-Net and MobileNetV3-SSDLite for underwater object detection is proposed. We deployed the network onto NVIDIA Xavier NX, and conducted experiments of underwater object detection. The experimental results show the advantage of proposed network in underwater object detection tasks;
- FPGA implementation of 1-stride and 2-stride convolution with 3×3 and 5×5 filter was optimized. Those convolutions were all implemented in a single engine to maximize the reuse of the multiplier in FPGA and improve the data processing capacity of the accelerator;
- An efficient engine for 2×2 upsampling is proposed. It only consumes 4/9 of the multipliers needed in the Winograd algorithm, and the upsampling speed is also faster. The FPGA implementation of SE module in MobileNetV3 is optimized by decomposing the computational procedure. A memory access is presented based on "ping-pong" operation to improve data reuse.

The rest of the article is organized as follows. Section 2 summarizes related work and motivation. Section 3 introduces the detail of proposed end-to-end underwater object detection network and the leading algorithm. The FPGA accelerator design for the proposed network is described in Section 4. Section 5 shows the results of underwater object detection experiments and the results of FPGA implementation and comparison. Section 6 gives the conclusion and future work.

2. Related Work

Previous studies have proposed many machine learning-based approaches for underwater object detection and classification. Oliver et al. [34] applied scale-invariant feature transform (SIFT) to underwater object detection based on the simulated object features. Pizarro et al. [35] proposed bag of features object recognition system based maximum likelihood classifier (MLC). Burguera et al. [36] and Conzalez-Cid et al. [37] performed underwater object classification with support vector machine (SVM). However, all these algorithms are shallow intelligent and too dependent on handcrafted features. With the development of deep learning, CNN-based algorithms have shown significant advantages in object detection tasks. Py et al. [38] and Dai et al. [39] presented their deep neural network (DNN) for plankton detection, but their models have a too high computational cost since they build the neural network with large size kernels. To overcome the class-imbalance problem of training data, Lee et al. [40] employed the transfer learning combined pre-training CNN and original data fine-tuning. Faster R-CNN was applied in different underwater object detection tasks [41,42]. VGGNet is used as corals classifier in work [43]. To achieve real-time object detection on remotely operated vehicle platform, Yao et al. [44] optimized the basic network of MobileNet-SSD. Sung et al. [45] used YOLO for fish detection and achieved 16.7 fps object detection speed in GPU. In work [46], YOLOv3 was implemented on NVIDIA Jetson TX2 and the experimental results showed that it can achieve real-time object detection. However, the above research does not take any effective approaches to enhance the quality of underwater images before performing object detection. This results in the specific imaging environment seriously affecting the accuracy of underwater object detection and restricting the practical development of underwater robots.

Compared to GPUs, FPGA are a more popular design platform for AUVs due to the conveniences of expanding peripheral interfaces and customizing special hardware control logic. For implementing CNNs onto FPGA, most of previous studies focused on improving the computational parallelism and data reuse. By applying data compression and maximizing local data reuse to reduce data movement, Eyeriss [17] and Eyeriss v2 [18] achieved high energy efficiency and improved the utilization of computation resources. Work [47] improved the computational parallelism by separating the convolution computation and other data processing such as pooling and full connection. Bai et al. [24] and Wu et al. [22] proposed specific CNN accelerators for implementing depth-wise separable convolution onto FPGA. However, these works did not use fast algorithms to reduce the computational cost of convolution operations. Wang et al. [48] presented corresponding fast convolution units for computing convolutions in the CNN models based on fast finite impulse response algorithm (FFA). Nevertheless, this work has weak flexibility to support various convolutions with different strides and kernel sizes. FFT is used in work [26]. Compared to FFT, Winograd algorithm is more efficient for convolution with small kernel sizes [27]. Podili et al. [28] optimized the FPGA implementation of VGG16 based on Winograd algorithm and achieved 142.3 ms overall latency. Kala et al. [29] proposed an accelerator based on Winograd and GEMM. However, this work did not optimize the 2-stride convolution. Besides, Winograd $F(6,3)$ had been used as the fundamental computation unit in this work. Such large transformation matrices result in complex pre-computation and more latency. Yang et al. [32] and Yepez and Ko [33] further applied the Winograd algorithm for 2-stride convolution by decomposing the input feature map titles and kernels.

Different to the above-mentioned object detection research of AUVs, the underwater images captured by the camera are restored by U-Net before performing object detection in our FPGA-based underwater robot. Works [4,5] showed that U-Net can achieve excellent performance in underwater images restoration. Since the state-of-the-art lightweight CNNs such as MobileNets [11–13] and ResNet [15] greatly relieve the pressure of implementing real-time object detection on edge computation platform, it is worth considering spending some extra computational resource to obtain better-quality underwater images for object detection. For the case of FPGA implementation, even optimizing the convolution based on Winograd algorithm, 3/4 of the multiplications are wasted when performing deconvolution-based upsampling in U-Net. Meanwhile, due to the limited on-chip memory, it needs to access the external memory frequently when implementing the SE module of MobileNetV3 on FPGA. This will significantly reduce the speed of object detection. Thus, a novel upsampling engine and a SE engine is presented in this paper. The proposed underwater object detection CNN network combined U-Net and MobileNetV3-SSDLite is successfully implemented into the FPGA of our underwater robot, where it can achieve real-time object detection.

3. Proposed Underwater Object Detection CNN Network and Leading Algorithm

To achieve a high object detection accuracy, it is necessary to restore the images before performing object detection. U-Net is a considerable option because of its lightweight architecture and excellent performance of underwater images restoration.

3.1. Proposed End-to-End Underwater Object Detection CNN Network

This paper proposes a CNN network combined U-Net and MobileNetV3-SSDLite. As shown in Figure 1, the raw underwater images captured by camera in underwater vehicles are first restored by U-Net, and then the reconstructed image will be fed to MobileNetV3-SSDLite for detecting object.

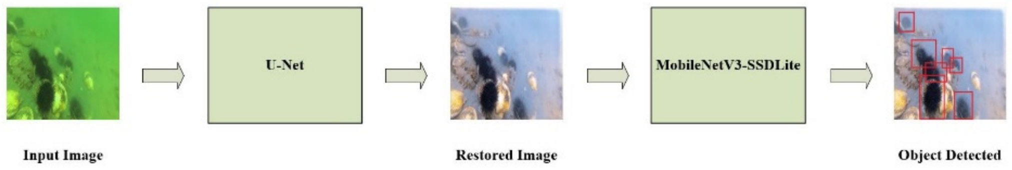


Figure 1. Proposed end-to-end underwater object detection CNN network.

3.1.1. Underwater Images Restoration Based on U-Net

Figure 2 shows the detailed description of U-Net in the proposed network. A 300 × 300 RGB underwater image will gradually be downsampled into a 38 × 38 × 256 vector through continuous convolutions and downsampling in the encoder part of U-Net. In each downsampling stage, 1-stride 3 × 3 standard convolution is conducted twice followed by a rectified linear unit (ReLU) activated function and a 2-stride 2 × 2 max-pooling. In the decoder part, 2 × 2 upsampling is first conducted of the input feature maps. After each upsampling operation, output feature maps are concatenated to the corresponding symmetric layer in the encoder side, and are then followed by two consecutive 3 × 3 standard convolutions and a ReLU activation layer.

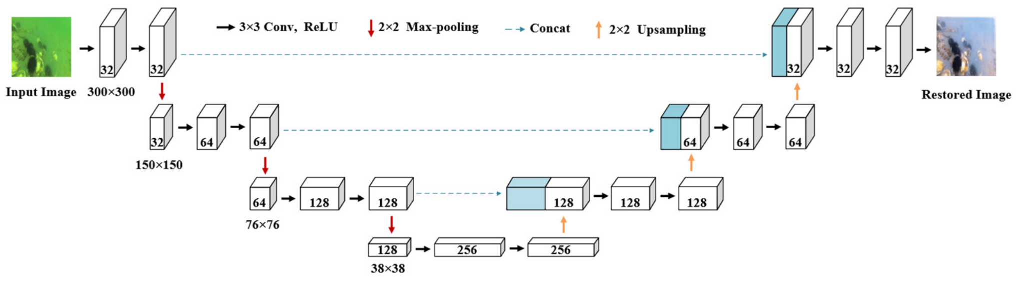


Figure 2. U-Net architecture for underwater image restoration.

3.1.2. MobileNetV3-SSLite Object Detection Model

The architecture of MobileNetV3-SSDLite is shown in Figure 3. The feature extractor is the base network of the MobileNetV3-SSDLite, followed by four extra convolutional layers which enable features to be extracted at multiple scales. For MobileNetV3-Large (M3_Large), C4 represents the expansion layer of the 13-th bottleneck block. For MobileNetV3-Small (M3_Small), C4 represents the expansion layer of the 9-th bottleneck block. For all feature extractor, C5 represents the layer immediately before pooling.

3.2. Leading Algorithm

MobileNets [11–13] are the representative lightweight CNN network. By adopting depth-wise separable convolution instead of standard convolution, MobileNets significantly reduce the number of parameters and computation. When implementing the trained CNN network, using a Winograd algorithm can further reduce the multiplication required for convolution.

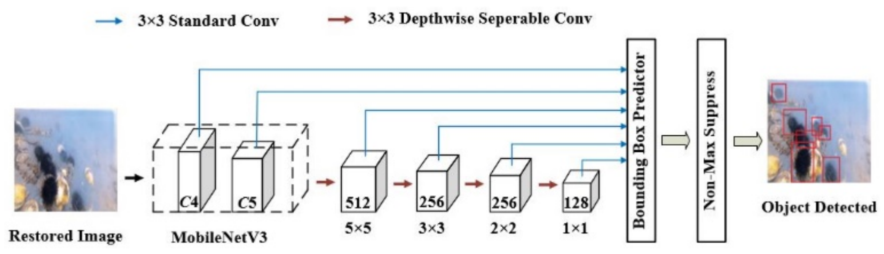


Figure 3. The network architecture of MobileNetV3-SSDLite.

3.2.1. Depthwise Separable Convolution and the Bneck of MobileNetV3

MobileNets [11–13] construct CNN based on depthwise separable convolution (DSC). As shown in Figure 4, by decomposing standard convolution (SC) into depthwise convolution (DWC) followed by pointwise convolution (PWC), DSC significantly reduces the parameters and computation of convolution. Considering input feature maps size $W \times H \times C_1$ and the kernels size $K \times K \times C_1 \times C_2$, compared with SC, the reduction factors on parameter and computation can be written as:

$$F_p = F_c = \frac{1}{C_2} + \frac{1}{K^2}. \tag{1}$$

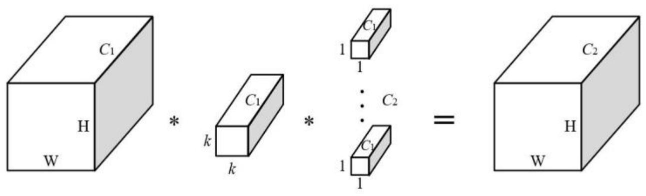


Figure 4. Depthwise separable convolution.

As the successor of previous versions, MobileNetV3 [13] further simplifies the network structure and applies SE module into the inverted residual block. The structure of the bottleneck in MobileNetV3 is shown in Figure 5. SE squeezes the DWC output feature maps into a $1 \times 1 \times C$ vector so-called channel descriptor through global average pooling, and generates a vector represented by the channel contribution weights of feature maps after performing two consecutive full-connected layers. The output vector of the SE module is used to do the channel-wise multiplication with original output feature maps of DWC, which could make the output feature maps of the bottleneck more high-quality. It is noted that SE is an optional module in some bottlenecks instead of exiting in all layers.

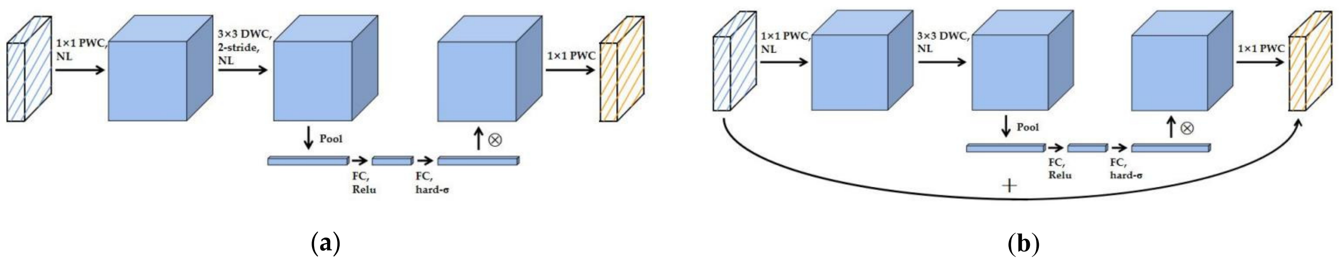


Figure 5. The structure of bottleneck with squeeze-and-excitation in MobileNetV3. (**a**) stride = 1 block, (**b**) stride = 2 block.

3.2.2. Winograd Algorithm

The Winograd algorithm [27], an efficient convolution decomposition model, is widely used to optimize DWC and SC. In the case of 2-D convolution with the stride of 1, the Winograd algorithm can reduce the number of convolutional multiplications from m^2r^2 to n^2. Assuming the input title d sizes $n \times n$, the filter kernel g sizes $r \times r$ and the output title Y sizes $m \times m$, where $m = n - r + 1$. Using Winograd algorithm $F(m \times m, r \times r)$, the convolution can be formulated as:

$$Y = A^T[(GgG^T)\cdot(B^T dB)]A, \tag{2}$$

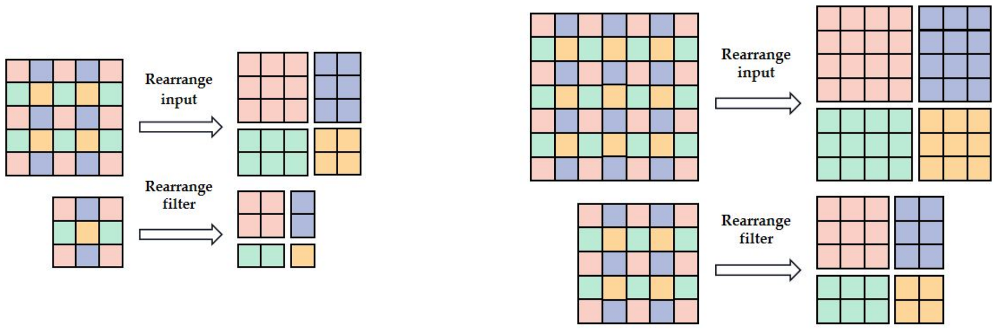
where, symbol "$\cdot$" represents element-wise matrix multiplication. It is noted that the transformation matrices A, B and G in $F(m \times m, r \times r)$ are the same as $F(m, r)$.

However, the original Winograd algorithm is not suitable to accelerate the convolution layers with non-1 stride. Refs. [32,33] proposed a strategy of decomposing 2-D 2-stride convolution, that is, decomposing and recombining the input feature title and convolution kernel according to the location of elements. Then, each decomposed feature sub-titles only do convolution with the corresponding decomposed sub-kernel. All the convolution results of those sub-titles are added up finally. As shown in Figure 6, 2-stride convolution with 3×3 filter can be implemented by accumulating the result of one $F(2 \times 2, 2 \times 2)$, four $F(2, 2)$ and one element-wise convolution with 4×4 input title; 2-stride convolution with 5×5 filter can be implemented by accumulating the result of one $F(2 \times 2, 3 \times 3)$, one $F(2 \times 2, 3 \times 2)$, one $F(2 \times 2, 2 \times 3)$ and one $F(2 \times 2, 2 \times 2)$. It is noted that $F(2 \times 2, 3 \times 2)$ can be implemented by considering the matrices of $F(2, 3)$ as A^T, G and B^T, and considering the matrices of $F(2,2)$ as G^T, B and A; $F(2 \times 2, 2 \times 3)$ can be implemented by considering the matrices of $F(2, 2)$ as A^T, G and B^T, and considering the matrices of $F(2, 3)$ as G^T, B and A. The transformation matrices of $F(2, 2)$, $F(2, 3)$ and $F(2, 5)$ can be respectively written as Equations (3)–(5):

$$B^T = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \\ 0 & -1 & 1 \end{bmatrix} \quad G = \begin{bmatrix} 1 & 0 \\ 1/2 & 1/2 \\ 1/2 & -1/2 \end{bmatrix} \quad A^T = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & -1 \end{bmatrix}, \tag{3}$$

$$B^T = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad G = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & 1/2 \\ 0 & 0 & 1 \end{bmatrix} \quad A^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix},. \tag{4}$$

$$A^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & 2 & -2 & 1 \end{bmatrix}$$
$$B^T = \begin{bmatrix} 4 & 0 & -5 & 0 & 1 & 0 \\ 0 & -4 & -4 & 1 & 1 & 0 \\ 0 & 4 & -4 & -1 & 1 & 0 \\ 0 & -2 & -1 & 2 & 1 & 0 \\ 0 & 2 & -1 & -2 & 1 & 0 \\ 0 & 4 & 0 & -5 & 0 & 1 \end{bmatrix} \quad G = \begin{bmatrix} 1/4 & 0 & 0 & 0 & 0 \\ -1/6 & -1/6 & -1/6 & -1/6 & -1/6 \\ -1/6 & 1/6 & -1/6 & 1/6 & -1/6 \\ 1/24 & 1/12 & 1/6 & 1/3 & 2/3 \\ 1/24 & -1/12 & 1/6 & -1/3 & 2/3 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$



(a) (b)

Figure 6. Decomposing 2-stride 2-D convolution to several 1-stride 2-D convolution. (**a**) convolution with kernel = 3 × 3, (**b**) convolution with kernel = 5 × 5.

4. FPGA Accelerator Design for Proposed Network

In this paper, some computational engines including Winograd-based convolution engine, 2×2 upsampling engine, and SE engine are proposed to improve computational parallelism. A customized memory access strategy is designed to balance the parallelism and memory bandwidth.

4.1. Hardware Architecture Overview

The accelerator consists of three functional modules: computing controller, on-chip memory control logic, and computing engines. The processing system (PS) activates the accelerator via AXI4-Lite, and then the *Computation Controller* will generate a series of instructions. According to the instructions, the *im_reader* in the on-chip memory control logic loads features maps titles, convolution kernel, and bias from external memory DDR to on-chip memory BRAM, and dispatches the data to the specified computing engine. *Winograd Convolution Engine* is the computing engine of DWC and SC in the proposed network. The 2×2 *Upsampling Engine*, *SE Engine*, and *PWC Engine* are used to calculate 2×2 Upsampling of U-Net, SE module of MobileNetV3 and PWC, respectively, and multiplier and adder resources can be shared between the engines. *Max-pooling Engine* is used to calculate max-pooling operation in U-Net. The computation results are written back into on-chip memory and are saved back to external memory by *im_writer*. Figure 7 gives an overview of the proposed FPGA accelerator.

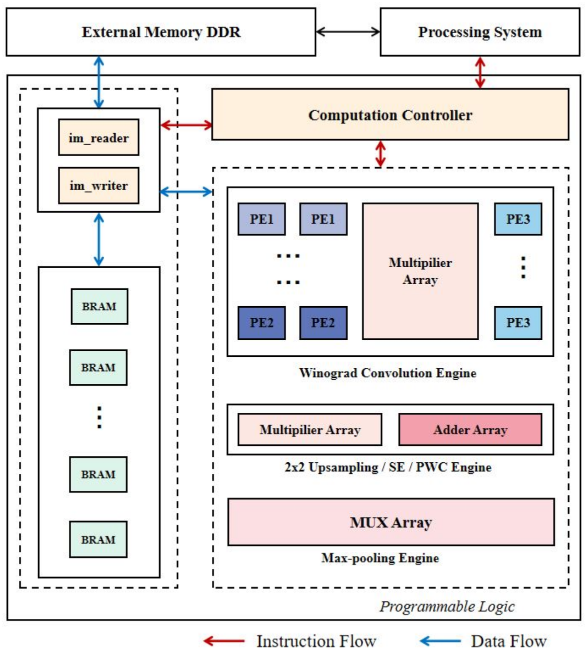


Figure 7. Accelerator FPGA design overview.

4.2. Winograd Convolution Engine

Winograd convolution engine shown in Figure 8, is used to compute 3×3 SC in U-Net and 1-stride and 2-stride 3×3, 5×5 DWC in MobileNetV3-SSDLite. $F(m \times m, r \times r)$, which is described in Equation (2), can be divided into four independent parts. Wherein, PE1 is used to transform the input feature map titles d into $B^T dB$, and convolution kernel g is transformed GgG^T into by PE2. Results of PE1 and PE2 are conducted hadamard product in DSP array, and hadamard product results z are transformed into $A^T zA$ by PE3. All PEs in the Winograd convolution engine are designed with 2-cycle pipeline. Specifically, all PEs perform pre-multiply of the transformation in the first cycle and post-multiply in the second cycle. Since transformation matrix A^T, G and B^T are known constant matrices, the matmul product of Winograd transformation in all PEs can be implemented by shifter and adder by doing an approximation (such as $1/6 \approx 1/8 + 1/32$). For a different convolution, PEs will dispatch the input feature map titles and kernels to the corresponding transformation module through DEMUX according to the instruction. In this paper, one Winograd engine contains 150 DSPs with the mode of $A \times B$. The output results of different rows of the Winograd convolution engine are first stored in different FIFOs by *im_writer*, and are then written back to the external memory to ensure that the output feature maps could correctly store in the continuous space.

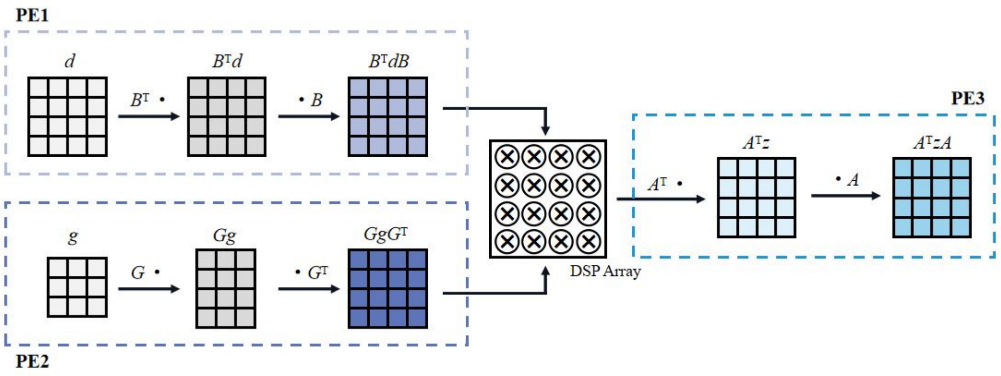


Figure 8. The structure of Winograd computational engine.

For the case of DWC calculation followed by ReLU nonlinear activation, batch normalization, or 2×2 downsampling, the accelerator would spend 3 to 6 cycles to process the output results of the Winograd convolution engine before storing these results in FIFO. Because the mean value and variance obtained during the training proposed network are constant when applied to object detection task, batch normalization can easily be implemented by the shifter and adder in a single cycle.

4.3. 2×2 Upsampling Engine

Upsampling in the U-Net is implemented by deconvolution. That is, a 22 upsampling needs to expand and pad feature maps with 0-value elements to double the spatial size of the input first, then perform 2×2 convolution with the stride of 1. However, due to the large number of inserted 0-value element, 5/9 of multiplications are wasted while applying Winograd 2×2 convolution to expanded feature maps.

An efficient upsampling engine shown in Figure 9 is presented in this paper. Since only the middle element of all input feature map titles of $F(2 \times 2, 2 \times 2)$ is the non-0 element, the upsampling can be converted to accumulate the results of element convolution of input feature map titles and transformed kernel, as described in Figure 9. Each upsampling engine contains 16 DSPs with the mode of $A \times B$, channel-wise accumulator, and a post-process module. To obtain the correct output feature maps, the post-process module is used to control the order of storing the accumulator output results into on-chip memory. The DSPs and adders in the accumulators of upsampling engines can be reused to implement *SE Engine* and *PWC Engine*. In the decoder part of U-Net, results of the last SC in each stage are sent to *Winograd Convolution Engine* to calculate 3×3 SC of the next stage after 2×2 upsampling instead of writing back to the external memory.

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & d_k & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = d_k * \begin{bmatrix} w_{22} & w_{21} \\ w_{12} & w_{11} \end{bmatrix} = \begin{bmatrix} d_k w_{22} & d_k w_{21} \\ d_k w_{12} & d_k w_{11} \end{bmatrix}. \tag{6}$$

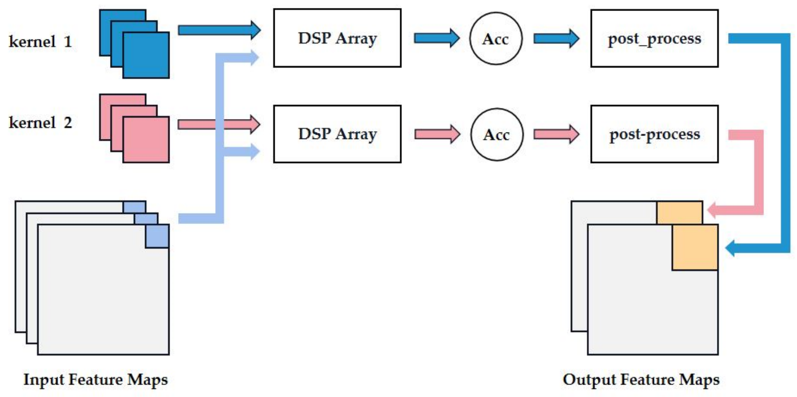


Figure 9. Upsampling engine.

4.4. SE Engine

The SE module is applied into the bottleneck of the partial layers of MobileNetV3. By reusing the multipliers and adders in 2 × 2 *Upsampling Engine*, the FPGA implementation of *Global Average Pooling* and the first full-connected (FC) layer in SE are optimized in this paper. The details are shown in Figure 10.

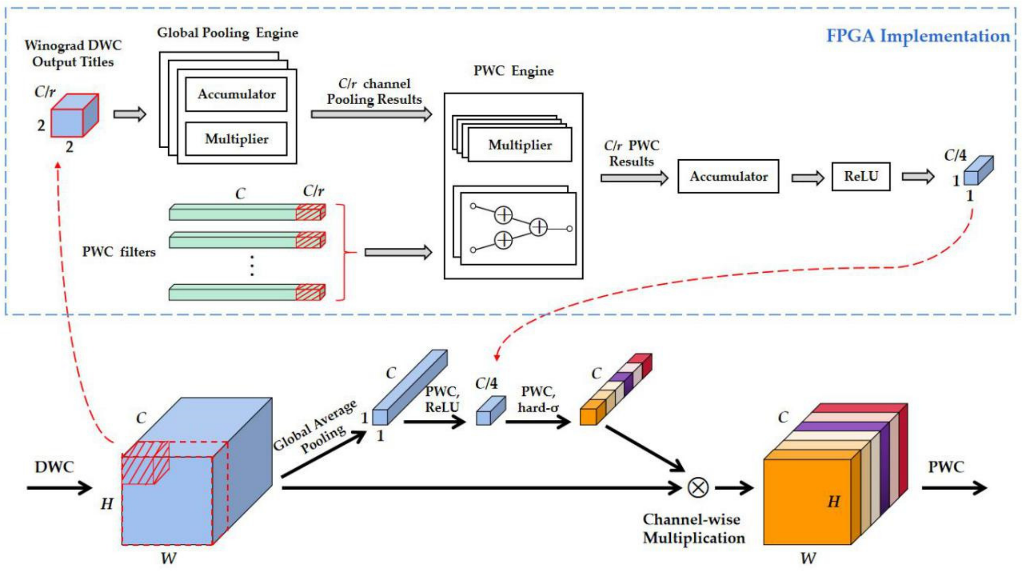


Figure 10. Squeeze-and-excitation in MobileNetV3.

(1) *Global Average Pooling*: One *Winograd Convolution Engine* generates an output feature map title, which sizes $2 \times 2 \times C/r$ (the value of C/r depended on the specific channel-wise parallel process strategy in different layers). The output title needs to be copied before being written back to the external memory. The copied data conducted channel-wise accumulation until the C/r channels feature maps had been processed, and the results are stored on the on-chip memory. Then, the accumulation results of the C/r channels feature maps is multiplied by a constant $1/HW$ to obtain the global pooling result.

(2) *PWC + ReLU*: The first FC layer in SE is implemented by PWC and ReLU. Instead of waiting until all the input feature maps had been pooled, the PWC of the first FC is performed when the global average pooling results of C/r channels input feature maps are obtained. The $1 \times 1 \times C/r$ channel description vector and corresponding kernel elements are sent into *PWC Engine*, then PWC results are assigned to the accumulator to be accumulated with the PWC results of the previous $1 \times 1 \times C/r$ channel description vector. ReLU nonlinear activation is performed after obtaining the PWC results of C channels input feature map.

The SE output a $1 \times 1 \times C$ vector (hereinafter referred as **v**) after two consecutive FC layers. **v** needs to be conducted in channel-wise multiplication with output feature maps of DWC in bneck of MobileNetV3, but the output feature maps are stored in the external memory when **v** is obtained. Therefore, the following strategy is adopted in this paper: **v** is saved on the on-chip memory, and the feature maps titles read from the external memory need to be multiplied with the corresponding elements in v before conducting the PWC of bneck.

4.5. Memory Access

Since the computation engine could not work until all the needed data contained feature maps titles and corresponding filter weights, the memory access delay has played a pivotal role in the optimization of the FPGA accelerator. To reduce the latency caused by data transfer among on-chip and external memory, we overlap data transfer and data computation. To be specific, *im_reader* in Figure 7 employs two groups of FIFOs to load the data. The second group memory receives the data while the computation modules fetch the data in the first group memory to compute, and next time the first group memory receive data while the computation modules fetch the data in the second group memory. A similar way is used to write back the computation results.

5. Experiments and Results Analysis

5.1. Experimental Setup

The proposed underwater object detection CNN network is implemented on NVIDIA Jetson Xavier NX and FPGA. Xavier NX contains 384-cores NVIDIA Volta GPU and 48 Tensor Cores. The FPGA is a Zynq XC7Z045 platform, which consists of a Kintex-7 FPGA and dual ARM Cortex-A9 processor. One GB DDR3 SDRAM integrated in the PS side is used as the external memory and 16-bit quantization strategy is chosen. The FPGA accelerator runs at a frequency of 150 MHz while the ARM processor runs at 800 MHz. Moreover, the bounding box predictors of MobileNetV3-SSDLite are implemented on PS side to overlap it with other computation on PL side, which is illustrated in Figure 11.

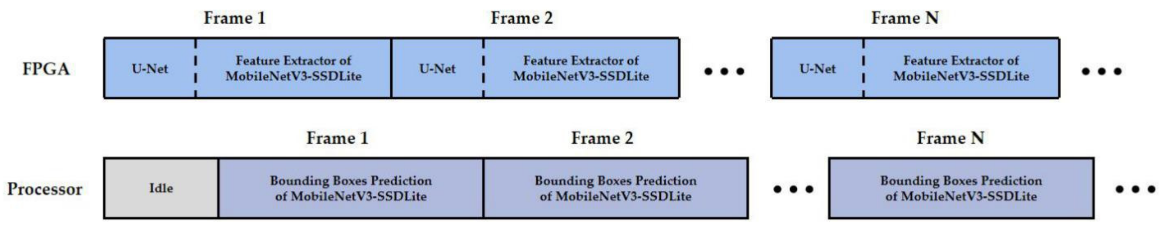

Frame 1
Frame 2
Frame N
FPGA
U-Net
Feature Extractor of MobileNetV3-SSDLite
U-Net
Feature Extractor of MobileNetV3-SSDLite
U-Net
Feature Extractor of MobileNetV3-SSDLite
Frame 1
Frame 2
Frame N
Processor
Idle
Bounding Boxes Prediction of MobileNetV3-SSDLite
Bounding Boxes Prediction of MobileNetV3-SSDLite
Bounding Boxes Prediction of MobileNetV3-SSDLite

Figure 11. Overlap of the operation in PL side and PS side.

The testing image data set is PASCAL VOC dataset, which consists of 3375 images collected from the underwater image open dataset Real-world Underwater Image Enhancement (RUIE) Dataset provided by Dalian University of Technology [49], including 1715 near-field green images and 1660 far-field blue-green images.

5.2. Underwater Object Detection Accuracy

As shown in Table 1, we collect the mAP of the proposed network using an IoU threshold from 0.5 to 0.75 with an interval of 0.05. In this paper, mAP@T is used to indicate the mAP calculated with IoU threshold T, and mmAP is used to indicate mAP averaged over all IoU values we collected. It can be found that no matter whether M3_Large or M3_Small is used as the feature extractor of MobileNetV3-SSDLite, the mAP of the underwater object detection with restored images is higher than that with the original underwater images under a different IoU threshold. As illustrated in Figure 12, with the increase of the IoU threshold, the mAP improvement of object detection between restored underwater images and original underwater images is a growing tendency. This shows that it is necessary to apply the U-Net to pre-process the underwater images in high-precision demand object detection.

Table 1. Object detection results on different IoU threshold based on PASCAL VOC dataset.

	mmAP (%)	mAP (%) @0.5	mAP (%) @0.55	mAP (%) @0.6	mAP (%) @0.65	mAP (%) @0.7
M3_Large-SSDLite	56.82	65.3	61.9	57.4	52.3	46.0
U-Net + M3_Large-SSDLite	63.34	71.6	68.4	64.3	59.4	53.4
M3_Small-SSDLite	46.67	55.4	51.8	47.6	42.3	36.2
U-Net + M3_Small-SSDLite	53.66	62.1	58.8	54.6	49.5	43.3

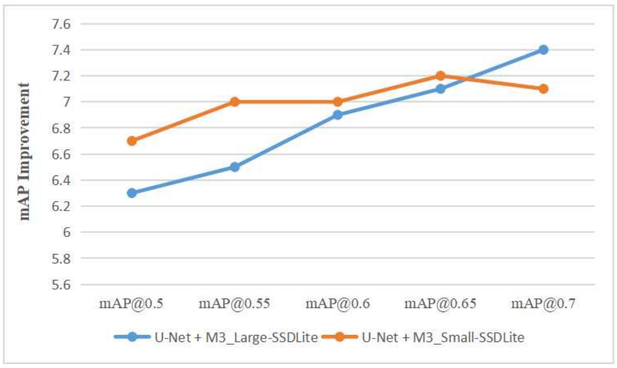


Figure 12. mAP improvement on a different IoU threshold of proposed underwater object detection CNN network with different extractor.

5.3. Performance Comparison

There is other rare yet similar research to this paper that focuses on implementing a CNN network combined with U-Net and MobileNetV3-SSDLite on FPGA. To compare the performance of our FPGA design and other platforms, we implemented the proposed network among Xavier NX and Intel i7-8700 16 GB RAM CPU based on a PyTorch framework [50]. CUDA and cuDNN are used for optimizing the GPU solution of Xavier NX. To make a fair comparison, we implemented the proposed CNN network with 32-bit float point precision and 16-bit float point precision, respectively, on Xavier NX and CPU. Table 2 shows the power consumption and performance of different platforms. Note that the Zynq XC7Z045 is a product from 2021 and Xavier NX is from 2019. The generation difference in production technology inevitably puts XC7Z045 at a disadvantage in performance comparison. Using more recent FPGA should result in better object detection performance.

Our proposed FPGA design achieves 23.68 fps, which is considered to be sufficient in many of the real-life underwater object detection demands of AUV. For the case of 16-bit float point precision, compared to CPU, the FPGA accelerator achieves a higher object detection speed (more than 7.5×) and a better energy efficiency (more than 52×) regardless of whether it uses M3_Large or M3_Small as the feature extractor. For a comparison with Xavier NX, our accelerator achieves about 0.64× energy efficiency and the object detection speed of our design is nearly 0.43× that of Xavier NX.

5.4. Design Complexity Comparison

It is hard to quantify the design complexity of the proposed underwater object detection CNN network on a different platform. There is no doubt that because of cuDNN and CUDA provided by NVIDIA, which enable users to accelerate CNN with GPU conveniently, Xavier NX has a significant advantage due to design complexity for many special applications such as implementing the CNNs model. However, although implementing and debugging algorithms onto FPGA is cumbersome, FPGA-based schemes are more efficient and offer low-complexity in many ways when it comes to designing a whole hardware system instead of only one object detection task for AUVs.

Table 2. Performance comparison of the CPU, GPU, and FPGA platform.

	CPU				**GPU**				**Our Work**	
Platform	Intel i7-8700				Xavier NX				Zynq XC7Z045	
Compiler	GCC 7.3.0				CUDA 10.2 cuDNN 8.0				Vivado 2019.2	
Frequency	3.2 GHz				1.2 GHz				150 MHz	
Power (W)	65 [1]				13.8				9.34	
Technology	14 nm				12 nm				28 nm	
Feature Extractor Model	M3_Large		M3_Small		M3_Large		M3_Small		M3_Large	M3_Small
Precision	32-bit float	16-bit float	32-bit float	16-bit float	32-bit float	16-bit float	32-bit float	16-bit float	16-bit fixed	16-bit fixed
Speed (fps)	1.47	2.71	2.32	4.39	28.1	54.8	40.22	79.01	23.68	33.14
Energy Efficiency (fps/W)	0.023	0.042	0.036	0.068	2.036	3.971	2.914	5.725	2.535	3.548

[1] The power of CPU is referenced from thermal design power (TDP) of its specification [51].

5.5. FPGA Implementation Results

Table 3 shows the resource utilization of our accelerator on Zynq XC7Z045 clocked at 150 MHz. Although the Winograd algorithm reduces the cost of DSPs per convolution, we used 89.1% of the DSPs on FPGA to achieve high throughput, which enables the accelerator to acquire a higher giga operations per second (GOPS). Correspondingly, 166 K LUTs are used in our design and most of them are used to implement the Winograd transformation mentioned in Section 4.2. A total of 81.8% of the available BRAM is used to store the input/output data and data reuse.

Table 3. FPGA resource utilization.

	LUT	**DSP**	**BRAM**	**FF**
Available	218,600	900	545	437,200
Used	166,403	801	446	159,141
Utilization	76.1%	89.1%	81.8%	36.4%

6. Conclusions and Future Work

This paper proposed a CNN network combined U-Net and MobileNetV3-SSDLite to achieve high performance underwater object detection. The proposed network improves the underwater object detection mAP under a multiple IoU threshold. To implement the proposed network on our FPGA-based underwater robot, we optimize the FPGA implementation of various convolution in the network, and propose an efficient upsampling PE. Moreover, the FPGA implementation of the squeeze-and-excitation module in MobileNetV3 is optimized in this paper.

The accelerator is implemented on the Xilinx Zynq XC7Z045 FPGA and runs at a clock frequency of 150 MHz. When using M3_Large as the feature extractor, the average processing speed of our accelerator is approximately 24 fps, which achieves 8.7× speedup and 60.4× energy efficiency compared to CPU and achieves 0.43× speedup but 0.64× energy efficiency compared to GPU. For the case of using M3_Small as the feature extractor, the object detection speed is about 33 fps, which achieves 7.5× speedup and 52.2× energy efficiency compared to CPU and achieves 0.42× speedup but 0.61× energy efficiency compared to GPU. Although there is a performance disparity in regards to processing speed and energy efficiency between our accelerator and advanced GPU, our accelerator is sufficient in the requirement of many FPGA-based customized underwater vehicles.

For future work, we plan to explore 12-bit and 10-bit quantization strategy into FPGA implementation. Besides, since nonlinear activation will result in sparse output feature maps, it is a large possibility on an implementation level to optimize convolution.

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

Funding: This research received no external funding.

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

References

1. Kim, B.; Yu, S. Imaging sonar based real-time underwater object detection utilizing AdaBoost method. In Proceedings of the 2017 IEEE Underwater Technology (UT), Busan, Korea, 21–24 February 2017; pp. 1–5.
2. Matteoli, S.; Corsini, G.; Diani, M.; Cecchi, G.; Toci, G. Automated Underwater Object Recognition by Means of Fluorescence LIDAR. *IEEE Trans. Geosci. Remote Sens.* **2015**, *53*, 375–393. [CrossRef]
3. Ancuti, C.; Ancuti, C.O.; Haber, T.; Bekaert, P. Enhancing underwater images and videos by fusion. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, 16–21 June 2012; pp. 81–88.
4. Drews, P., Jr.; Nascimento, E.D.; Moraes, F.; Botelho, S.; Campos, M. Transmission Estimation in Underwater Single Images. In Proceedings of the 2013 IEEE International Conference on Computer Vision Workshops, Sydney, Australia, 3–6 December 2013; pp. 825–830.
5. Fabbri, C.; Islam, M.J.; Sattar, J. Enhancing Underwater Imagery Using Generative Adversarial Networks. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 7159–7165.
6. Hashisho, Y.; Albadawi, M.; Krause, T.; Lukas, U.F.V. Underwater Color Restoration Using U-Net Denoising Autoencoder. In Proceedings of the 2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA), Guangzhou, China, 10–13 May 2019; pp. 117–122.
7. Li, J.; Skinner, K.A.; Eustice, R.M.; Johnson-Roberson, M. WaterGAN: Unsupervised Generative Network to Enable Real-Time Color Correction of Monocular Underwater Images. *IEEE Robot. Autom. Lett.* **2018**, *3*, 387–394. [CrossRef]
8. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525.
9. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
10. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 8–10 June 2015; pp. 1–9.
11. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient convolutional neural networksfor mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
12. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
13. Howard, A.; Sandler, M.; Chu, G.; Chen, L.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for MobileNetV3. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 1314–1324.
14. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1800–1807.
15. Guo, Y.; Li, Y.; Wang, L.; Rosing, T. Depthwise Convolution Is All You Need for Learning Multiple Visual Domains. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 8368–8375.
16. Li, H.; Fan, X.; Jiao, L.; Cao, W.; Zhou, X.; Wang, L. A high performance FPGA-based accelerator for large-scale convolutional neural networks. In Proceedings of the 2016 26th International Conference on Field Programmable Logic and Applications (FPL), Lausanne, Switzerland, 29 August–2 September 2016; pp. 1–9.
17. Chen, Y.; Krishna, T.; Emer, J.S.; Sze, V. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. *IEEE J. Solid-State Circuits* **2017**, *52*, 127–138. [CrossRef]
18. Chen, Y.; Yang, T.; Emer, J.; Sze, V. Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2019**, *9*, 292–308. [CrossRef]
19. Ma, Y.; Cao, Y.; Vrudhula, S.; Seo, J. Optimizing the Convolution Operation to Accelerate Deep Neural Networks on FPGA. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2018**, *26*, 1354–1367. [CrossRef]
20. Qiu, J.; Song, S.; Yu, W.; Yang, H.; Xu, N. Going deeper with embedded FPGA platform for convolutional neural network. In Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), Monterey, CA, USA, 21–23 February 2016; pp. 26–35.

21. Guo, K.; Sui, L.; Qiu, J.; Yu, J.; Wang, J.; Yao, S.; Han, S.; Wang, Y.; Yang, H. Angel-Eye: A Complete Design Flow for Mapping CNN Onto Embedded FPGA. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2018**, *37*, 35–47. [CrossRef]
22. Wu, D.; Zhang, Y.; Jia, X.; Tian, L.; Li, T.; Sui, L.; Xie, D.; Shan, Y. A High-Performance CNN Processor Based on FPGA for MobileNets. In Proceedings of the 2019 29th International Conference on Field Programmable Logic and Applications (FPL), Barcelona, Spain, 8–12 September 2019; pp. 136–143.
23. Fan, H.; Liu, S.; Ferianc, M.; Ng, H.; Que, Z.; Liu, S.; Niu, X.; Luk, W. A Real-Time Object Detection Accelerator with Compressed SSDLite on FPGA. In Proceedings of the 2018 International Conference on Field-Programmable Technology (FPT), Dublin, Ireland, 27–31 August 2018; pp. 14–21.
24. Bai, L.; Zhao, Y.; Huang, X. A CNN Accelerator on FPGA Using Depthwise Separable Convolution. *IEEE Trans. Circuits Syst. II Express Briefs* **2018**, *65*, 1415–1419. [CrossRef]
25. Kurzak, J.; Tomov, S.; Dongarra, J. Autotuning GEMM Kernels for the Fermi GPU. *IEEE Trans. Parallel Distrib. Syst.* **2012**, *23*, 2045–2057. [CrossRef]
26. Abtahi, T.; Shea, C.; Kulkarni, A.; Mohsenin, T. Accelerating Convolutional Neural Network With FFT on Embedded Hardware. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2018**, *26*, 1737–1749. [CrossRef]
27. Lavin, A.; Gray, S. Fast Algorithms for Convolutional Neural Networks. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 4013–4021.
28. Podili, A.; Zhang, C.; Prasanna, V. Fast and efficient implementation of Convolutional Neural Networks on FPGA. In Proceedings of the 2017 IEEE 28th International Conference on Application-specific Systems, Architectures and Processors (ASAP), Seattle, WC, USA, 10–12 July 2017; pp. 11–18.
29. Kala, S.; Jose, B.R.; Mathew, J.; Nalesh, S. High-Performance CNN Accelerator on FPGA Using Unified Winograd-GEMM Architecture. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2019**, *27*, 2816–2828. [CrossRef]
30. Yang, T.; Liao, Y.; Shi, J.; Liang, Y.; Jing, N.; Jiang, L. A Winograd-Based CNN Accelerator with a Fine-Grained Regular Sparsity Pattern. In Proceedings of the 2020 30th International Conference on Field-Programmable Logic and Applications (FPL), Gothenburg, Sweden, 31 August–4 September 2020; pp. 254–261.
31. DiCecco, R.; Lacey, G.; Vasiljevic, J.; Chow, P.; Taylor, G.; Areibi, S. Caffeinated FPGAs: FPGA framework For Convolutional Neural Networks. In Proceedings of the 2016 International Conference on Field-Programmable Technology (FPT), Xi'an, China, 7–9 December 2016; pp. 265–268.
32. Yang, C.; Wang, Y.; Wang, X.; Geng, L. WRA: A 2.2-to-6.3 TOPS Highly Unified Dynamically Reconfigurable Accelerator Using a Novel Winograd Decomposition Algorithm for Convolutional Neural Networks. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2019**, *66*, 3480–3493. [CrossRef]
33. Yepez, J.; Ko, S. Stride 2 1-D, 2-D, and 3-D Winograd for Convolutional Neural Networks. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2020**, *28*, 853–863. [CrossRef]
34. Oliver, K.; Hou, M.; Wang, S. Feature matching in underwater environments using sparse linear combinations. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition—Workshops, San Francisco, CA, USA, 13–18 June 2010; pp. 60–67.
35. Pizarro, O.; Rigby, P.; Johnson-Roberson, M.; Williams, S.B.; Colquhoun, J. Towards image-based marine habitat classification. In Proceedings of the OCEANS 2008, Quebec City, QC, Canada, 15–18 September 2008; pp. 1–7.
36. Burguera, A.; Bonin-Font, F.; Lisani, J.L.; Petro, A.B.; Oliver, G. Towards automatic visual sea grass detection in underwater areas of ecological interest. In Proceedings of the 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), Berlin, Germany, 6–9 September 2016; pp. 1–4.
37. Gonzalez-Cid, Y.; Burguera, A.; Bonin-Font, F.; Matamoros, A. Machine learning and deep learning strategies to identify Posidonia meadows in underwater images. In Proceedings of the OCEANS 2017—Aberdeen, Aberdeen, UK, 19–22 June 2017; pp. 1–5.
38. Py, O.; Hong, H.; Zhongzhi, S. Plankton classification with deep convolutional neural networks. In Proceedings of the 2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference (ITNEC 2016), Chongqing, China, 20–22 May 2016; pp. 132–136.
39. Dai, J.; Wang, R.; Zheng, H.; Ji, G.; Qiao, X. ZooplanktoNet: Deep convolutional network for zooplankton classification. In Proceedings of the OCEANS 2016—Shanghai, Shanghai, China, 10–13 April 2016; pp. 1–6.
40. Lee, H.; Park, M.; Kim, J. Plankton classification on imbalanced large scale database via convolutional neural networks with transfer learning. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 3713–3717.
41. Xia, Y.; Satter, J. Visual Diver Recognition for Underwater Human-Robot Collaboration. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 6839–6845.
42. Li, X.; Shang, M.; Hao, J.; Yang, Z. Accelerating fish detection and recognition by sharing CNNs with objectness learning. In Proceedings of the OCEANS 2016—Shanghai, Shanghai, China, 10–13 April 2016; pp. 1–5.
43. Mahmood, A.; Bennamoun, M.; An, S.; Sohel, F.; Boussaid, F.; Hovey, R.; Kendrick, G.; Fisher, R.B. Coral classification with hybrid feature representations. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 519–523.

44. Yao, Y.; Qiu, Z.; Zhong, M. Application of improved MobileNet-SSD on underwater sea cucumber detection robot. In Proceedings of the 2019 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chengdu, China, 20–22 December 2019; pp. 402–407.
45. Sung, M.; Yu, S.; Girdhar, Y. Vision based real-time fish detection using convolutional neural network. In Proceedings of the OCEANS 2017—Aberdeen, Aberdeen, UK, 19–22 June 2017; pp. 1–6.
46. Liu, S.; Li, X.; Gao, M.; Cai, Y.; Rui, N.; Li, P.; Yan, T.; Lendasse, A. Embedded Online Fish Detection and Tracking System via YOLOv3 and Parallel Correlation Filter. In Proceedings of the OCEANS 2018 MTS/IEEE Charleston, Charleston, SC, USA, 22–25 October 2018; pp. 1–6.
47. Desoli, G.; Chawia, N.; Boesch, T.; Singh, S.; Guidetti, E.; Ambroggi, F.D.; Majo, T.; Zambotti, P.; Ayodhyawasi, M.; Singh, H.; et al. 14.1 A 2.9TOPS/W deep convolutional neural network SoC in FD-SOI 28nm for intelligent embedded systems. In Proceedings of the 2017 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 5–9 February 2017; pp. 238–239.
48. Wang, J.; Lin, J.; Wang, Z. Efficient Hardware Architectures for Deep Convolutional Neural Network. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2018**, *65*, 1941–1953. [CrossRef]
49. Media Lab of Dalian University of Technology, X.; Dalian University of Technology, X. Realworld Underwater Image Enhancement RUIE Benchmark. Available online: https://github.com/dlut-dimt/Realworld-Underwater-Image-Enhancement-RUIE-Benchmark (accessed on 19 October 2021).
50. PyTorch, X. PyTorch. Available online: https://pytorch.org (accessed on 19 November 2021).
51. Intel, X. Intel® Core^{TM} i7-8700B Processor (12M Cache, up to 4.60 GHz). Available online: https://www.intel.com/content/www/us/en/products/sku/134905/intel-core-i78700b-processor-12m-cache-up-to-4-60-ghz/specifications.html (accessed on 19 November 2021).