Optimization of Deep-Learning Detection of Humans in Marine Environment on Edge Devices

Mostafa Rizk, Dominique Heller, Ronan Douguet, Amer Baghdadi, Jean-Philippe Diguet

Optimization of Deep-Learning Detection of Humans in Marine Environment on Edge Devices

M. Rizk[†§], D. Heller[‡], R. Douguet[‡], A. Baghdadi[†] and J-Ph. Diguet[◇]
[†]*IMT Atlantique, Lab-STICC UMR CNRS 6285, Brest, France*
[‡]*Universtité de Bretagne-Sud, Lab-STICC UMR CNRS 6285, Lorient, France*
[◇]*CNRS, IRL CROSSING, Adelaide, Australia*
[§] *Lebanese International University, CCE Department, Lebanon*

***Abstract*—Artificial intelligence (AI) detection techniques based on convolution neural networks (CNNs) require high computations and memory. Their deployment on embedded edge devices, with reduced resources and power budget, is highly hindered especially for applications that requires real-time inference. Several optimization methods such as pruning, quantization and using shallow networks, are mainly utilized to overcome this limitation but at the cost of degradation in detection performance. However, efficient approaches for training and inference have been recently introduced to lower such degradation. This work investigates the use of these approaches to optimize the popular You Only Look Once (YOLO) network targeting various emerging edge devices (Nvidia Jetson Xavier AGX, AMD-Xilinx Kria KV260 Vision AI Kit, and Movidius Myriad X VPU) in order to enhance the detection of humans in maritime environment.**

***Index Terms*—Marine, Object detection, Deep Learning, YOLO, Optimization**

I. Introduction

Convolutional Neural Network (CNN) architectures have been widely adopted to handle the challenge of detecting multiple objects in images and videos. Real-life applications impose the requirements of accuracy and precision as well as insuring real-time detection. These requirements come with a high cost in terms of computational resources at training and inference phases. The implementation of CNN-based architectures on embedded edge devices with limited available energy and resources introduces a major challenge. Several solutions have been introduced in this context. These solutions include methods of adopting light networks , pruning and quantization.

Human detection in marine environments is an important application for several disciplines such as marine search and rescue missions, man overboard accidents and illegal marine immigration. Recent studies show that marine incidents and irregular migration have led to increased number of losses in human lives [1][2]. Deep learning (DL) techniques based on CNNs have been adopted to detect humans in several application domains. However, few works tackle the detection of floating humans in open water using DL.

Recently, You Only Look Once (YOLO) [3] has been introduced as an efficient unified model of all phases of a CNN for detecting multiple objects in real-time. Several versions of YOLO exist, with different light networks suitable for embedded systems and low-power modes. However, these networks differ in their detection performance and inference rate. In this work, we aim to investigate these models in detecting floating humans and examine their performance in terms of accuracy, precision, and detection speed. In addition, the impact of optimization techniques, in particularly quantization and pruning, is evaluated when deployed on emerging embedded edge devices such as Nvidia Jetson Xavier AGX, AMD-Xilinx Kria KV260 Vision AI Kit, and Movidius Myriad X VPU.

This work was supported in part by the Regional Council of Bretagne through the ODESSA FEDER project.

The rest of the paper is organized as following. Section II presents a brief background and reviews related work. Section III describes the adopted methods and presents the obtained results along with the discussion. Finally, Section V concludes the paper.

II. Background and Related Work

A. Optimization techniques of DNNs on low resource devices

Deep Neural Networks (DNNs) have advanced significantly over the past few years in a variety of applications in the field of computer vision. In particular, CNN architectures have been improved to address the challenging tasks of image classification and object detection. The evolution of CNNs have introduced a solution to detect several objects in images and videos with acceptable accuracy and precision which are considered on par with human performance. In fact, deep learning models are traditionally dense and over-parameterized. This over-parameterization benefits the convergence of gradient descent during training but comes at the cost of additional memory and computation effort during model training and inference.

To cope with the challenges of implementing CNNs on embedded edge devices with limited computational resources and power budget, several optimization approaches have been introduced in the literature. The common techniques includes: (1) using of light models of the networks with fewer layers and reduced parameters, (2) pruning to reduce the number of parameters, (3) and quantization of weight and/or activation values. These techniques reduce the impact of the required computations and memory but at the cost of accuracy and precision.

In this paper, we explore these techniques for the application of human detection in marine environment. For network model design, we target the original YOLOv4 network and the compressed network YOLOv4 Tiny with both *MISH* and *leakyRelu* activation layers [4]. YOLOv4 network consists of 162 layers based on CSPDarknet53 structure. YOLOv4 Tiny uses the simplified network structure of CSPDarknet53-tiny. It compromises 38 layers and only two detector heads. For parameter quantization, half floating-point precision (FP16) and INT8 inference is adopted targeting embedded edge platforms. Parameter pruning is applied on YOLOv4 model to reduce the memory utilization and computational needs on target edge devices. In particular, we make use of sparsification to reduce the complexity by zeroing out subsets of the model parameters [5]. The impact of these optimization techniques are investigated in terms of detection performance and inference rate.

B. Human detection in marine environment using deep learning on edge devices

Most of the available works on object detection in maritime environments using deep learning target the detection of ships [6]. CNN-based techniques have been used to detect humans in several

recent works targeting multitude of applications such as crowd detection, security, search and rescue missions [7][8][9][10][11]. However, few works have focused on human detection in maritime scenes [12][13][14][15].

The performance of Faster R-CNN in detecting floating persons has been investigated in [13] while using thermal images. In this work, the authors have not indicated the target device. Only the training and testing results are presented. Man overboard event detection from RGB and thermal images is performed using single shot detector (SSD) and YOLOv3 in [12]. However, this work lacks to the performance results in terms on accuracy, precision and detection speed as well as the used device. The authors in [14], have deployed YOLOv3 to detect and localize human in marine environment using images captured by drones for search and rescue missions. The used dataset includes only 450 areal images captured in the same location. Also, the inference speed on embedded device is not addressed. The authors show the training and testing performances using their dataset. In [15], the authors have deployed YOLOv3 Tiny on NVIDIA Jetson TX1 to detect floating persons captured in areal images in the context of search and rescue missions using UAVs. This work shows that trained YOLOv3 Tiny model can achieve 12 FPS detection of humans in open water with mean average precision (mAP) of 67% evaluated on validation dataset.

III. Experiments and Results

A. Dataset

The used dataset contains 6462 images of humans in marine scenes from various positions and from different perspectives and scales with different backgrounds, resolutions and luminosity. The images are collected from previous published datasets [15] and other internet resources. YOLO_Mark tool is used for marking bounded boxes of humans or for adjusting the existing bounding boxes of labeled images to meet with the dimensions of the persons. Overall, 16795 bounding boxes are created. Note that the number of humans varies among the collected images. The collected images are split randomly by 70% as training dataset, 10% as validation dataset and 20% as testing dataset.

B. Training

The original structures of the target networks are not modified. The depth size of the last convolution layers connected to the layers of the detection layers are modified to fit with one class. In order to preserve the generalization, transfer learning is opted starting from models trained on COCO dataset with 80 classes. Before launching the training, the weights of the detection and classification layers are deleted; whereas, those of feature extraction layers are maintained. Upon training on our dataset, all weights are updated. The number of images per batch is adjusted to 64. The total number of iterations is set to 20000. The learning rate for training is initially fixed to 0.001 and it is set to be scaled down by the factor of 0.1 at iteration 16000 and iteration 18000. The input images are down sampled into resolutions (Res) of 416×416 or 608×608. Data augmentation (DA) is applied during the training process. Several DA modes are applied such as mosaic, cutmix, rotation and changing exposure and saturation.

C. Evaluation

The validation of the models is performed while training using the validation dataset. The mAP is computed for each 4 epochs based on the AP50 metric defined in the MS COCO competition. Fig. 1 illustrates the training and validation performances. Note that the blue curves correspond to the training losses; whereas, the red curves correspond to the computed mAP values. Testing of the trained models is conducted using the testing dataset. Table. I shows the obtained results in terms of popular object detection metrics.

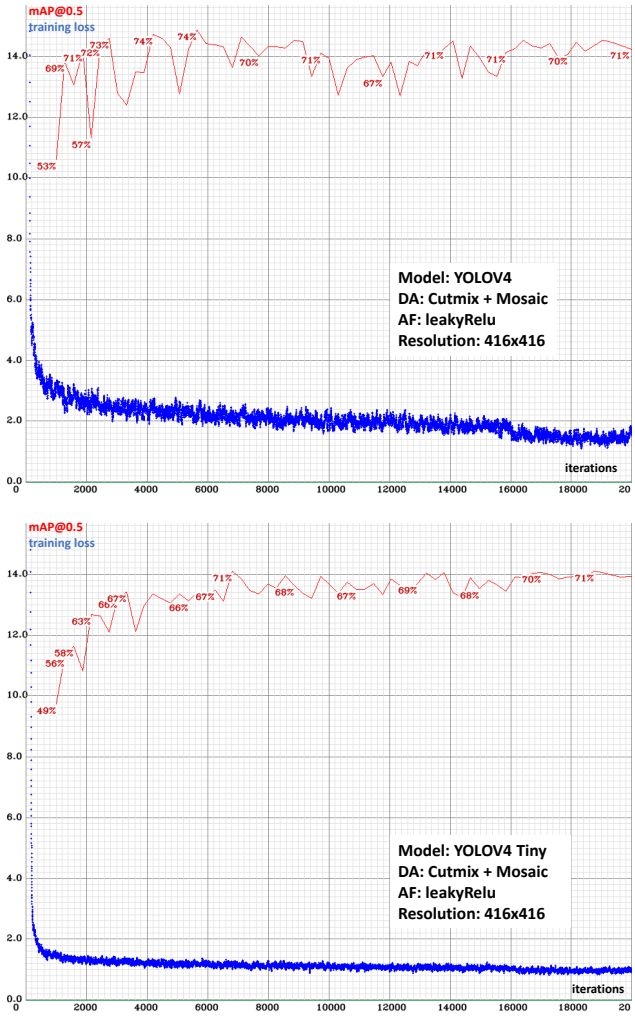

Fig. 1. Sample training and validation performances

TABLE I
Evaluation results of the trained YOLOv4 models

Target Model	AF	Resolution	DA	Precision	Recall	F1 score	Avr IOU	mAP @0.50
YOLOv4	*MISH*	416×416	without	0.71	0.53	0.60	50.08%	0.513
			with	0.77	0.74	0.76	55.32%	0.728
		608×608	without	0.71	0.43	0.54	50.28%	0.43
			with	0.78	0.73	0.75	55.92%	0.686
	LeakyRelu	416×416	without	0.59	0.33	0.42	41.30%	0.326
			with	0.77	0.73	0.75	55.66%	0.712
		608×608	without	0.70	0.27	0.39	48.93%	0.302
			with	0.75	0.71	0.73	53.50%	0.675
YOLOv4 Tiny	*MISH*	416×416	without	0.70	0.73	0.72	50.05%	0.693
			with	0.70	0.74	0.72	49.61%	0.691
		608×608	without	0.72	0.74	0.73	50.71%	0.719
			with	0.72	0.74	0.73	50.71%	0.719
	LeakyRelu	416×416	without	0.71	0.73	0.72	49.94%	0.697
			with	0.70	0.74	0.72	49.61%	0.691
		608×608	without	0.73	0.73	0.73	51.32%	0.711
			with	0.69	0.74	0.71	47.52%	0.681

D. Structured Pruning / Sparsifying

In order to achieve high-speed inference with high-precision detection, YOLOv4 model is pruned at the level of channels and layers. First, training under channel-level sparsity-induced regularization is performed in order to identify insignificant channels [16]. L1 regularization of the loss during training is adopted based on the work presented in [17]. The cost function is modified by adding a penalty term on the scaling factor (sparse weights). Also, a parameter λ is used to balance the normal training loss and the penalty term defined on the weights. Different values of λ are examined using the collected dataset to determine the best value. To determine whether the sparseness is sufficient, we use the Guppy multiple moving averages (Gmma) weight distribution map of each batch normalization (BN) layer. During sparsity training, it can be noticed that Gmma weights tend to close to zero indicating more sparseness as shown in Fig. 2.

Channel pruning is then applied to eliminate the channels with little contribution by deleting its input-output connections and corresponding weights. In this step, channels with near-zero scaling factors are pruned using a global threshold across all layers. A specific percentile of all the scaling factor values is used to define this threshold. In order to achieve the best value of the global threshold, we used the strategy of large intervals and then gradually subdividing to approach the optimal pruning point. In this work, 92% pruning leads to the optimal pruning point.

After channel pruning, layer pruning is performed in order to address the cross layer connections (residual) in YOLOv4 network where the output of a layer is the input to several subsequent layers. The previous CBL (Conv + Batch Normalizaton + Leaky-Relu) of each shortcut layer in the network is evaluated. Accordingly, the Gmma mean of each layer is sorted and the smallest layer is chosen for layer pruning. To determine the best number of *Resunits* to be cut, several experiments have been conducted while evaluating the obtained accuracy based on the metrics of precision, recall, mAP and F1-score. In this work, the best choice to maintain the accuracy of the model as much as possible is to cut 20 *Resunits* which imposes the removing of 48 layers in total.

Fine tuning is then applied to assist the pruned model to restore accuracy. The pruned compact model is retrained for 300 epochs. Fig. 3 shows the obtained performance metrics while fine-tuning the compact model.

Table. II presents the obtained results after each pruning step of YOLOv4 network with *leakyRelu* activation function. The table shows that the mAP drops by 2.3 points after sparsity training. However, this degradation, which is due to the modification of the loss function, is compensated later by the conducted fine-tuning on the pruned network. Also, the results illustrate that the channel pruning greatly reduces the number of model parameters (-98.4%) and the FLOPS (-91.64%) that involve a speed-up effect on embedded devices without suffering from accuracy loss. Fine-tuning of the YOLOv4 network recovers the accuracy loss due sparsity training. The obtained results show that the fine-tuned network achieves higher accuracy ($+2.1$ points in mAP) than the baseline network while preserving a significant decrease in the required memory (-90.4% reduction of parameters) and computations (-91.66% reduction of FLOPS) .

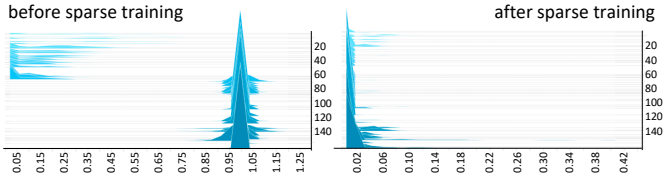

Fig. 2. Gmma weight distribution map of the BN layer

TABLE II
PRUNING RESULTS OF YOLOV4 NETWORK WITH *leakyRelu*

Step	Precision	Recall	mAP	F1-score	Parameters	BFLOPS
baseline	0.666	0.741	0.7	0.701	63937686	59.54
sparsity training	0.616	0.748	0.677	0.676	63937686	59.54
channel pruning	0.581	0.752	0.658	0.655	1037301	4.978
layer pruning	0.609	0.729	0.656	0.663	1028025	4.965
fine-tuning	0.643	0.78	0.721	0.705	1028025	4.965

E. Deploying trained model on edge devices

This subsection presents the deployment of the trained YOLOv4 models on Nvidia Jetson Xavier AGX, Xilinx Kria FPGA KV260 and Intel Movidius Myriad X Vision Processing Unit (VPU) integrated in OAK-D camera kit.

The deployment flow includes two basic steps:

1) quantization step where the bit-width of weights and activations is reduced to the desired representation using heuristic method and a golden reference pool of images selected from the training dataset.
2) compilation step in which the the model is optimized based on target hardware and then mapped into optimized instruction sequence ready to be deployed.

1) Deployment on Nvidia Jetson Xavier AGX: TensorRT is utilized in order to achieve lower latency and higher throughput inference on Jetson Xavier AGX. The target models are first converted to Open Neural Network Exchange (ONNX) and then to TensorRT engine with FP32 representation. Next, the target models are quantized to FP16 representation. Table. III presents the obtained detection performance and inference speed of the original trained models and the converted models using TensorRT when deployed on the Jetson Xavier AGX. The comparison shows that using TensorRT increases the inference rate while achieving better accuracy for all tested networks. This refers to the optimization process that implements several techniques such as kernal fusion, precision calibration, kernel auto-tuning, dynamic tensor memory and multi-stream execution. The quantization leads to better inference rate of 52 FPS (+247%) but a cost of degradation in the detection performance (-2.9 points in mAP).

2) Deployment on Xilinx Kria KV260: The trained models on darknet framework are first converted to a frozen TensorFlow graphs as Vitis AI does not support the graph provided by darknet framework. Also, the *MISH* activation layers are not supported by the Deep Learning Processor Unit (DPU) in Xilinx FPGA. Therefore, the deployment of the models with *leakyRelu* activation functions is considered. Using Vitis AI, the transferred model is quantized into INT8 representation, which is the only supported one, and compiled targeting DPUCZDX8G architecture. Note that the Vitis AI compiler does not support slice operator, which is used in the YOLOv4 Tiny model, for the targeted DPU. So, this architecture model is modified accordingly. Table. IV presents the obtained results in terms of detection performance and inference speed of the deployed models on Kria KV260. The pruned YOLOv4 model can achieve the highest inference speed (69 FPS) on KV260 kit while the unpruned YOLOv4 network can achieve the best mAP with 1% degradation due to quantization. YOLOv4 Tiny can achieve an inference rate of 65 FPS while loosing 5% in the mAP when quantized to INT8. Note that the FPS values for FP32 representation are not listed in Table. IV as the DPU in Kria KV260 kit can run only values in INT8 representation.

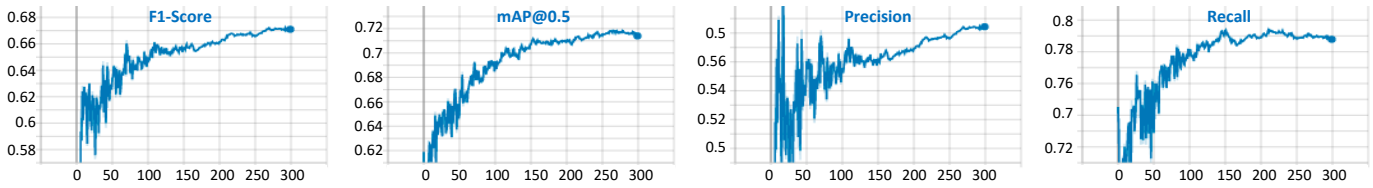

Fig. 3. Obtained performance metrics while fine-tuning the compact model

TABLE III
OBTAINED RESULTS ON JETSON XAVIAR AGX

Network	**YOLOv4 Tiny**			**YOLOv4**					
AF	*leakyRelu*			*leakyRelu*			*MISH*		
Resolution	**416×416**			**416×416**			**608×608**		
Model	original	FP32	FP16	original	FP32	FP16	original	FP32	FP16
mAP @0.5	0.8	0.818	0.795	0.859	0.861	0.830	0.815	0.826	0.802
FPS	90	92	115	15	20	52	9	10	30

TABLE IV
OBTAINED RESULTS ON KRIA KV260

Network	**YOLOv4 Tiny**		**YOLOv4**		**Pruned YOLOv4**	
Resolution	**416×416**		**416×416**		416 × 416	
Model	FP32	INT8	FP32	INT8	FP32	INT8
mAP @0.5	0.509	0.451	0.829	0.818	0.372	0.348
FPS	-	65	-	11	-	69

3) Deployment on Movidius Myriad X VPU: The pruned YOLOv4 model is also deployed on Movidius Myriad X VPU. Movidius Myriad X VPU is programmable with the Intel distribution of the OpenVINO [18]. The model is optimized and quantized to FP16 representation. The resultant model achieves an inference speed of 29.8 FPS (2 threads running on 6 SHAVE cores) and 14.7 FPS (1 thread running on 6 SHAVE cores) with a slight degradation of 1.2% in the mAP (71.01%) when compared to the unquantized pruned network (72.4%). In addition, the YOLOv4 Tiny model is deployed on Movidius Myriad X VPU. It achieves 30 FPS inference speed with a mAP of 72.4%.

F. Analysis

The analysis of the obtained results shows that Jetson Xavier AGX can achieve best inference speed and mean average precision but at the cost of higher power consumption. The power consumption of the GPU (30W) on Jetson Xavier AGX is 3.75 times more than KRIA KV260 AI VISION KIT (8W) and 7.5 times more that of Movidius Myriad X VPU (4W). The performance per Watt is determined when running the models on all targeted embedded edge platforms. Jeston Xavier AGX can achieve 3.83 FPS/W and 1.73 FPS/W for YOLOv4 Tiny and YOLOv4 networks respectively. Kria KV260 Vision AI kit can achieve 8.125 FPS/W, 1.375 FPS/W and 8.625 FPS/W for YOLOv4 Tiny, YOLOv4 and pruned YOLOv4 networks. Movidius Myriad X VPU can achieve 7.45 FPS/W for pruned YOLOv4 model and 7.5 FPS/W for YOLOv4 Tiny model.

IV. CONCLUSION

In this paper, we have investigated the deployment of YOLO on edge devices for efficient human detection in maritime environments. Three recent emerging edge devices have been considered in this study: Nvidia Jetson Xavier, AMD-Xilinx Kria KV260 Vision AI Kit, and Movidius Myriad X VPU. Channel, layer and fine pruning techniques, together with different levels of quantization, have been applied to enable high detection speed without scarifying accuracy and precision. Furthermore, the impact of different parameters such as network model, activation function, image resolution and data augmentation has been analyzed. The proposed deployments demonstrate promising results with an inference speed of 50 FPS and limited degradation of 2.9% in mAP.

REFERENCES

[1] E. M. S. Agency, "Annual overview of marine casualties and incidents 2021," EMSA, Annual Report, Dec. 2021.
[2] "International Organization for Migration Missing Migrants Project website," http://missingmigrants.iom.int, accessed: 2022-05-01.
[3] J. Redmon *et al.*, "You Only Look Once: Unified, real-time object detection," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.
[4] A. Bochkovskiy *et al.*, "YOLOv4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934
[5] T. Hoefler *et al.*, "Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks," *Journal of Machine Learning Research*, vol. 22, no. 241, pp. 1–124, 09 2021.
[6] D. Qiao *et al.*, "Marine vision-based situational awareness using discriminative deep learning: A survey," *Journal of Marine Science and Engineering*, vol. 9, no. 4, 2021.
[7] G. Castellano *et al.*, "Preliminary evaluation of TinyYOLO on a new dataset for search-and-rescue with drones," in *International Conference on Soft Computing Machine Intelligence (ISCMI)*, 2020, pp. 163–166.
[8] C. Liu and T. Szirányi, "Real-time human detection and gesture recognition for on-board UAV rescue," *Sensors*, vol. 21, no. 6, 2021.
[9] M. Rizk *et al.*, "Toward AI-assisted UAV for human detection in search and rescue missions," in *2021 International Conference on Decision Aid Sciences and Application (DASA)*, Sakheer, Bahrain, Dec. 2021, pp. 781–786.
[10] S. Sambolek and M. Ivasic-Kos, "Automatic person detection in search and rescue operations using deep cnn detectors," *IEEE Access*, vol. 9, pp. 37 905–37 922, 2021.
[11] R. L. Rosero *et al.*, "Deep learning with real-time inference for human detection in search and rescue," in *Intelligent Systems Design and Applications*, A. Abraham *et al.*, Eds. Cham: Springer International Publishing, 2021, pp. 247–257.
[12] I. Katsamenis *et al.*, "Man overboard event detection from RGB and thermal imagery: Possibilities and limitations," in *Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, ser. PETRA '20. New York, NY, USA: Association for Computing Machinery, 2020.
[13] V. A. Feraru *et al.*, "Towards an autonomous UAV-based system to assist search and rescue operations in man overboard incidents," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Abu Dhabi, UAE, Nov. 2020, pp. 57–64.
[14] L. Qingqing *et al.*, "Towards active vision with UAVs in marine search and rescue: Analyzing human detection at variable altitudes," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Abu Dhabi, UAE, Nov. 2020, pp. 65–70.
[15] E. Lygouras *et al.*, "Unsupervised human detection with an embedded vision system on a fully autonomous UAV for search and rescue operations," *Sensors*, vol. 19, no. 16, 2019.
[16] M. Tian *et al.*, "Pruning-based YOLOv4 algorithm for underwater gabage detection," in *Chinese Control Conference (CCC)*, 2021, pp. 4008–4013.
[17] Z. Liu *et al.*, "Learning efficient convolutional networks through network slimming," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2755–2763.
[18] M. H. Ionica and D. Gregg, "The movidius myriad architecture's potential for scientific computing," *IEEE Micro*, vol. 35, no. 1, pp. 6–14, 2015.