Marine Objects Detection Using Deep Learning on Embedded Edge Devices

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

Marine Objects Detection Using Deep Learning on Embedded Edge Devices

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

***Abstract*—Artificial Intelligence techniques based on convolution neural networks (CNNs) are now dominant in the field of object detection and classification. The deployment of CNNs on embedded edge devices targeting real-time inference sets a challenge due to the limited computing resources and power budgets. Several optimization techniques such as pruning, quantization and use of light neural networks enable the real-time inference but at the cost of precision degradation. However, using efficient approaches to apply the optimization techniques at training and inference stages enable high inference speed with limited degradation of detection performance. In this paper, we revisit the problem of detecting and classifying maritime objects. We investigate different versions of the You Only Look Once (YOLO), a state-of-the-art deep neural network, for real-time object detection and compare their performance for the specific application of detecting maritime objects. The trained YOLO networks are efficiently optimized targeting three recent edge devices: Nvidia Jetson Xavier AGX, AMD-Xilinx Kria KV260 Vision AI Kit, and Movidius Myriad X VPU. The proposed deployments demonstrate promising results with an inference speed of 90 FPS and a limited degradation of 2.4% in mean average precision.**

***Index Terms*—Marine, Deep learning, Embedded edge devices, YOLO, Ship detection, Ship dataset, Optimization**

I. Introduction

Marine object detection and classification are two essential tasks for many applications such as vessel identification and positioning, collision avoidance system, safe autonomous navigation, search and rescue mission, etc. Marine objects can span from stationary floating objects such as buoys to small boats and kayaks and other large vessels such as ferries, passenger ships and cargo ships. Surveillance adopting shared information from Electronic Chart Display and Information System (ECDIS) and Global Navigation Satellite System (GNSS) can provide locations of marine vessels [1]. However, this depends on the reliability of data, which may degrade due to spoofing, jamming or even dis-activating automatic identification system (AIS). Radar-based methods can be effective to detect the presence of large vessels. Small boats and floating objects on water surface are difficult to be identified [1].

Visual detection of marine objects using electro-optical sensors provides a solution for detecting and classifying marine objects [2]. Classification and detection of objects using captured images have been widely used in several application domains [3][4]. However, the characteristics of the scenes captured in marine environment arise additional challenges to the task of detecting objects in images or videos compared to other environments. Factors such as dynamic nature of the background, unavailability of static cues, presence of small objects at distant backgrounds and illumination effects impact the performance of commonly used image processing and computer vision approaches [5]. Tides and waves lead to a continuously dynamic background in both spatial and temporal dimensions. Also, floating objects are subjected to a lot of motion with unpredictable patterns. The illumination of marine scenes varies due to weather conditions (haze, fog, rain, bright sunlight, twilight, etc.). Speckles and glints are mainly induced by the variation of the solar incident angle on water. Furthermore, the disparity of color gamut depends on illumination conditions. Color gamut varies respectively between dark, yellow and red, blue and gray during night, sunset, daylight, and hazy conditions. These factors affect the visibility of objects in marine environments and hence the detection performance. An effective technique for certain cases with specific illumination type, weather condition and water dynamicity may not suit other conditions.

Artificial intelligence (AI) techniques based on deep learning provide robust solutions to detect and locate objects. The achieved performance proves the relevance of convolution neural networks (CNNs) in circumventing existing computer vision challenges. Deep learning methods, known as deep neural networks, make use of multiple hidden layers between the input and output layers to learn a hierarchy of features that are invariant to geometric transformations from raw input images.

In this work, we investigate the use of deep neural networks to detect and classify marine objects. The state-of-the-art You Only Look Once (YOLO) networks are adopted to ensure high detection performance and high detection rate. Also, we make use of emergent optimization techniques to deploy the trained networks on embedded edge devices. The main contributions of this paper are:

- create a diverse dataset of marine objects, which has the widest number of classes and annotations compared to available datasets,
- train and evaluate several YOLO neural networks with different sizes and architecture specifications,
- apply structured pruning using sparsifying to reduce the network size while maintaining the detection performance,
- optimize the trained networks towards implementation on popular edge devices to achieve the best compromise between inference speed and detection performance.

The rest of the paper is organized as follows. Section II presents a brief background and reviews related work. Section III describes the adopted methods for training and evaluation. Section IV illustrates the used structured pruning method. Section

This work was supported in part by the Regional Council of Bretagne (through ODESSA and SURCOUF projects) and the City of Lorient (through SURCOUF project).

979-8-3503-9851-9/22/$31.00 ©2022 IEEE

V describes the deployment of models on edge devices and shows the obtained results. Finally, Section VI concludes the paper.

II. Background and Related Work

A. Object detection methods

Previously, computer vision techniques based on feature extraction has been widely adopted to detect objects [6][7]. Recently, AI methods with CNNs play dominant role in classifying and locating multi objects in images and videos leading to accurate detection. One-stage detection methods have been introduced to provide real-time performance with acceptable precision and accuracy. These methods exclude the stage of pre-selecting the regions of classification and abstract post-processing techniques (refining bounding boxes, eliminating duplicates and adjusting detection scores) used in two-stage methods such as the well known region-convolutional neural network (R-CNN) [8] and its enhanced versions [9], [10] in order to reduce the complexity and ensure real-time detection speed.

You Only Look Once (YOLO) has been recently proposed in [11] as an efficient one-stage CNN-based model that is able to detect multi objects in real-time. Published peer reviewed comparisons [12][1] [13] illustrate that YOLO outperforms two-stage detection methods and other available one-stage methods such as single shot detector (SSD). Since introduced, many versions of YOLO have been introduced such as YOLOv2[14], YOLOv3 [15], YOLOv4 [16] and and YOLOv5. In YOLOv2 [14], the fully connected layers at the end have been eliminated and Darknet-19 architecture has been adopted. YOLOv3 [15] uses Darknet-53 architecture and inherits the concept of residual networks. The detections are made at 3 different scales which enables the detection of small objects. YOLOv4 and YOLOv4-tiny proposed initially in 2020 [16] optimize and improve every part of YOLOv3. The main optimization is to use CSPDarknet-53 as its backbone network for extracting features. The difference between YOLOv4-tiny and YOLOv4 is that the tiny version only has two YOLO heads at the end (2 scale factor instead of 3).

The experiments targeting Microsoft Common object in context (COCO) dataset [17] show that YOLOv4 is faster and more accurate than real-time neural networks EfficientDet [18] and RetinaNet [19] provided by Google and Facebook respectively. Comparisons have been made between YOLOv3, YOLOv4 and YOLOv5, in which some authors claim that YOLOv4 is more accurate while others claim that YOLOv5 is more accurate. The reason for different reported results can be attributed to many factors, such as the different datasets used, the modified hyperparameters, etc [20].

B. Maritime Detection Datasets

Object detection based on deep learning imposes the challenge of having sufficient dataset with object annotations for training and validation processes. Training using large datasets with diverse images results in robust networks and prevents the occurrence of overfitting or underfitting. In maritime context, only few datasets for maritime object detection are available publicly for research purposes. This subsection presents briefly these available published datasets.

VAIS dataset [21] provides visible and infrared maritime images for ship classification [22][23]. It includes more than 1,000 paired RGB and infrared images among six ship categories (merchant, sailing, passenger, medium, tug, and small). The large-scale MARitime VEsseLs (MARVEL) dataset [24] has been introduced for classification of maritime vessels [25]. Although the dataset includes 2M images, the provided ground truth (GT) annotations contain the url to download image and the class label without bounding boxes. Also, the available scripts to retrieve the images from *ShipsSpotting* website [26] is no longer functioning. Both VAIS and MARVEL datasets are not applicable for deep learning detection.

Singapore Maritime Dataset (SMD) has been published in [3]. It compromises 81 videos, which are recorded using cameras either placed on-shore or fixed on-board of a moving vessel during daytime and nighttime. Also, the dataset includes thermal videos captured using a camera equipped by Near-IR bandpass filter. SMD annotations include 10 classes of maritime objects (ferry, buoy, vessel/ship, speed boat, boat, kayak, sail boat, swimming person, flying bird/plane, and *Other*). The GT labels for every frame of every video is provided comprising bounding-boxes and object classes for the corresponding bounding-box. This dataset has been evaluated in many research works such as [27] and [28]. Recently, an improvement of SMD called SMD-Plus is described in [29].

SeaShips [30] is another published dataset for detection. It consists of 31,455 images including 40,077 annotations of six categories of marine ships (ore ships, bulk carriers, general cargo ships, container ships, fishing ships and passenger ships). The images are extracted from 10,080 video segments recorded by surveillance cameras placed in the coastline. Bovcon et al. have introduced the MODS dataset in [31] for unmanned surface vehicles (USV) obstacle detection. This dataset merges MODD1 [32], MODD2 [33] and SMD [3] datasets. Overall, the MODS dataset contains 24,090 images and 145,334 annotated objects. MaSTr1325 dataset [34] is published as a marine semantic segmentation training dataset for the application of obstacle detection in small-sized coastal USVs. MODS and MaSTr1325 datasets are used in the context of obstacle detection; thus, the objects are categorized into two classes only: small obstacle and large obstacle.

In [35], *Mcships* dataset is introduced for ship detection and fine-grained categorization. The dataset includes 14,709 annotated images compromising 26,529 instances of ships which are categorized into 6 classes of warships and 7 classes of civilian ships. ABOships [36] is a recent publish dataset which consists of 9880 on-shore and off-shore images of maritime objects. Images in this dataset are categorized into boat, cargoship, cruiseship, ferry, militaryship, miscboat, miscellaneous, motorboat, passengership, sailboat, seamark.

III. Training and Evaluation

A. Collected Dataset

A diverse dataset of marine objects is created. Initially, we make use of the published datasets such as SMD [3], SeaShips [30] and MODD [32] datasets. These datasets include images with high level of similarity as the images are extracted from recorded video sequences. This may impose over-fitting while training the neural network. Therefore, the datasets are cleaned by removing images that are too similar and bad images. Also, the annotations are checked and adjusted by fixing the imprecise bounding boxes and wrong labels. The merging of these datasets

TABLE I
SPECIFICATIONS OF THE TARGETED YOLOV4 MODELS

Model	Number of Layers	activation function	Model Weights' Volume (MB)
YOLOv4	162	MISH	256.2
YOLOv4 Tiny	38	LeakyRelu	23.5

leads to 10K images split into 15 classes: Ferry, Buoy/SeaMark, sailing-vessel, tug, speed boat, kayak, bulk carrier, roro cargo, small boat, swimming person, flying bird, container ship, fishing vessel, passenger ship and jetski.

In addition to the images in the published datasets, we add new images of available classes and create new classes as well. The added images have different backgrounds, size, weather condition, light luminosity and show object from different points of view. The added images are taken from AIS providers [1]. To annotate new images, semi-automatic annotation is used. First, a neural network is trained using the images of public datasets. The trained network is then used to locate automatically the bounding boxes of marine objects in the new images.

During this step, images which are too similar in the same class are removed. Similar classes are merged to ovoid confusion while detection. This process is done attentively. For each iteration, a neural network is trained on the collected images considering all classes. Validation is done using images not seen by the trained model. A confusion matrix that represents the predictions in terms of a ground truth is created. Accordingly, classes with high confusion are merged.

As a result, our dataset includes 27,781 images compromising 35,140 annotations of 41 classes. Fig. 1 demonstrates the distribution of annotations per class of our dataset. The resulting dataset is the hugest available dataset for marine object detection in terms of number of classes and number of annotations. Note that the resulting dataset will be released to research community under request to facilitate progress in detection of marine objects using deep learning methods.

The dataset point of view is challenging due to two main reasons. First, in general, ships are very similar in shape (low inter-class variation). Second, due to viewpoint, weather condition and illumination variations, but also due to scale changes, occlusion, cluttered background and so on, ships within the same class may be significantly different (large intra-class variation) [35].

B. Target Models

In this work, YOLOv4 and YOLOv4 Tiny models are targeted. Table I presents the specifications of the targeted models. The depth of the layers at the input of YOLO layers in the networks are adjusted to fit with the number of classes. To meet with the supported architectures on Kria KV260 Vision AI kit, the activation function of YOLOv4 is modified from *MISH* to *LeakyRelu*. Also, the *maxpool* sizes of the spatial pyramid pooling (SPP) are changed to 3×3, 5×5 and 7×7 as the Deep Learning Processor Unit (DPU) has a maximum kernel size of 8×8.

C. Training

The training processes are conducted using *Darknet* framework [37] on Nvidia Quadro RTX 3090. Transfer learning is adopted by initiating the training with the weight values of models previously trained on popular large-scale COCO dataset. Only the weights of the detector layers are cleared. This act preserves the generalization of the feature extraction layers in the output model. During training the number of batches is set to 32. The model is trained for 75 epochs. The training rate is adjusted to 0.001. A scaling factor of 0.1 is applied at iterations 65600 and 73800. The input images are down sampled into resolutions of 416×416 or 608×608. Several Data augmentation (DA) modes are applied during the training process such as mosaic, cutmix, rotation and changing exposure and saturation.

D. Evaluation of trained models

The trained models are validated while training using the validation dataset. The mean average precision (mAP) is computed for each 4 epochs based on the AP50 metric defined in the MS COCO competition. Fig. 2 presents the training and validation performances. The blue curve in the figure corresponds to the training loss; whereas, the red curve corresponds to the computed mAP values. Fig. 3 shows sample predictions using the trained model.

In addition, the models are evaluated using several popular metrics in the field of object detection. Table II presents the obtained evaluation results for all trained models.

IV. STRUCTURED PRUNING USING SPARSIFYING

Pruning of the trained model at the level of channels and layers results in high-speed inference along with high-precision detection. Before starting the pruning, insignificant channels are identified using training under channel-level sparsity-induced regularization [38]. L1 regularization of the loss during training is adopted based on the work presented in [39]. 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. In this case $\lambda = 0.0015$ is chosen. 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. 4.

Accordingly, channel pruning is performed to eliminate the channels with little contribution. The input-output connections and the corresponding weights of these channels are deleted. 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. To attain the most relevant value of the global threshold, the strategy of large intervals is adopted followed by subdividing gradually the intervals to achieve the optimal pruning point. In this work, the optimal pruning point is reached for 78% pruning.

Layer pruning is then performed in order to address the cross layer connections (residual) in YOLOv4 network. These special types of connections link the output of one layer to the input of 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 saved and the layer with the smallest value is selected for pruning. Many experiments are performed in order to find the most adequate

[1]MarineTraffic: https://www.marinetraffic.com/

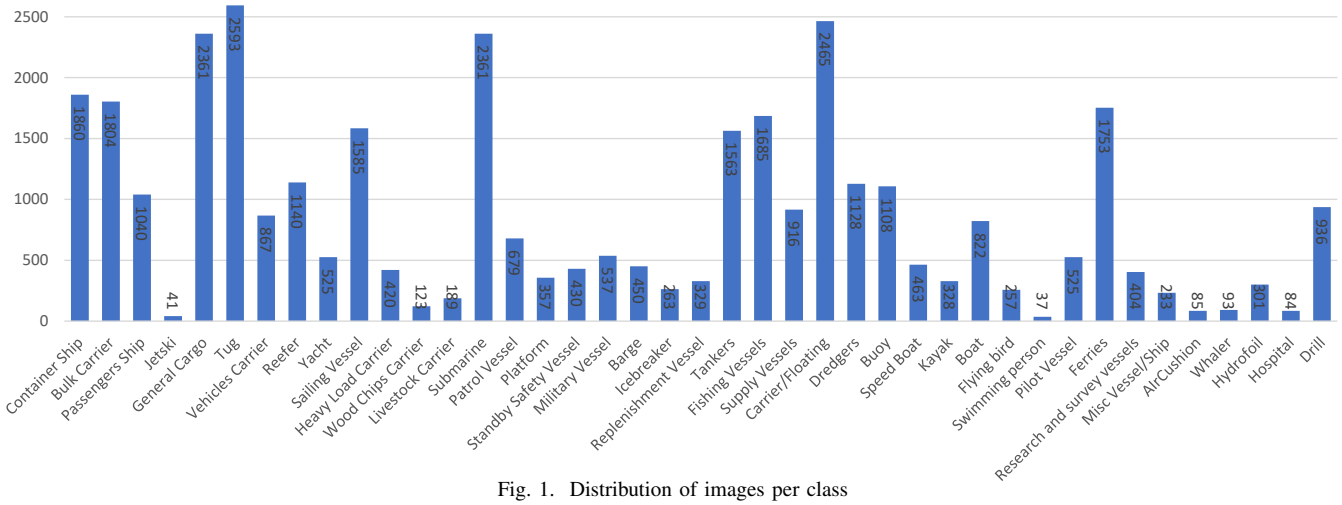

Fig. 1. Distribution of images per class

TABLE II
EVALUATION RESULTS OF THE TRAINED YOLOV4 MODELS

Target Model	Activation Function	Image resolution	Data augmentation	mAP COCO	mAP VOC07	mAP VOC12	Precisison	Recall	F1 score	Avg IOU
YoloV4 Tiny	LeakyRelu	416×416	-	0.625	0.613	0.626	0.55	0.70	0.61	42.50%
			Cutmix+mosaic	0.781	0.760	0.783	0.75	0.82	0.82	62.56%
YOLOV4	MISH	416×416	mosaic	0.838	0.813	0.840	0.81	0.86	0.84	71.56%
			Cutmix+mosaic	0.822	0.795	0.8250	0.81	0.86	0.83	71.48%
YOLOV4	MISH	608×608	mosaic	0.839	0.811	0.842	0.82	0.87	0.84	72.85%
			Cutmix+mosaic	0.838	0.812	0.841	0.82	0.87	0.84	72.61%
YOLOV4 without SPP	LeakyRelu	416×416	mosaic	0.827	0.806	0.829	0.82	0.86	0.83	72.39%
YOLOV4 with SPP	LeakyRelu	416×416	mosaic	0.831	0.808	0.833	0.82	0.86	0.84	73.68%

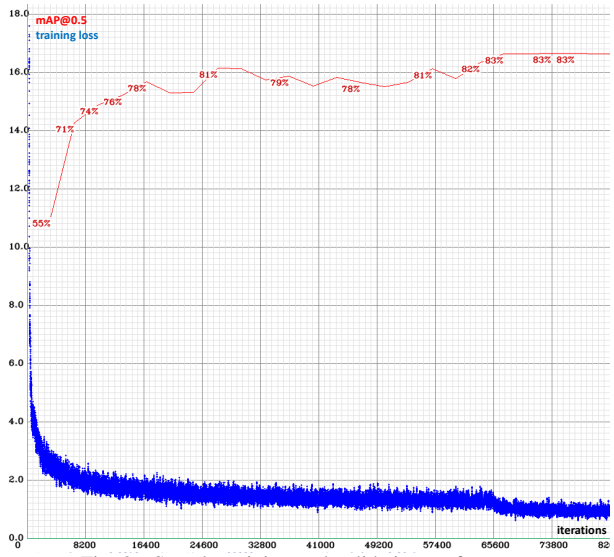

Fig. 2. Sample training and validation performances

number of *Resunits* to be removed. According to the evaluation of resultant accuracy based on the metrics of precision, recall, mAP and F1-score of all experiments the best choice is to cut 20 *Resunits* which imposes the removing of 48 layers in total. At last, fine tuning is then performed to assist the pruned model to restore accuracy. The pruned compact model is trained again for 300 epochs.

Table. III illustrates the obtained results after each pruning step of YOLOv4 network with *leakyRelu* activation function. The table

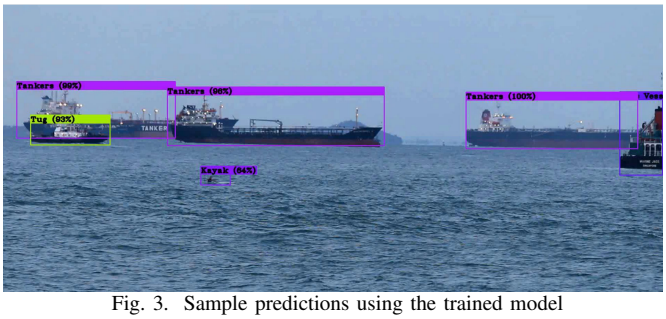

Fig. 3. Sample predictions using the trained model

shows that the mAP drops by 19.3 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 $(-97,06\%)$ and the FLOPS (-88.87%) 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 to sparsity training. The obtained results illustrate that the fine-tuned network achieves a mAP of 77% while preserving a significant decrease in the required memory $(-97,08\%$ reduction of parameters) and computations $(-88,88\%$ reduction of FLOPS). Fig. 5 shows the obtained performance metrics while fine-tuning the compact model.

V. DEPLOYING TRAINED MODELS ON EDGE DEVICES

Deep learning-based object detection technology in embedded systems needs to be optimized for low latency and high ac-

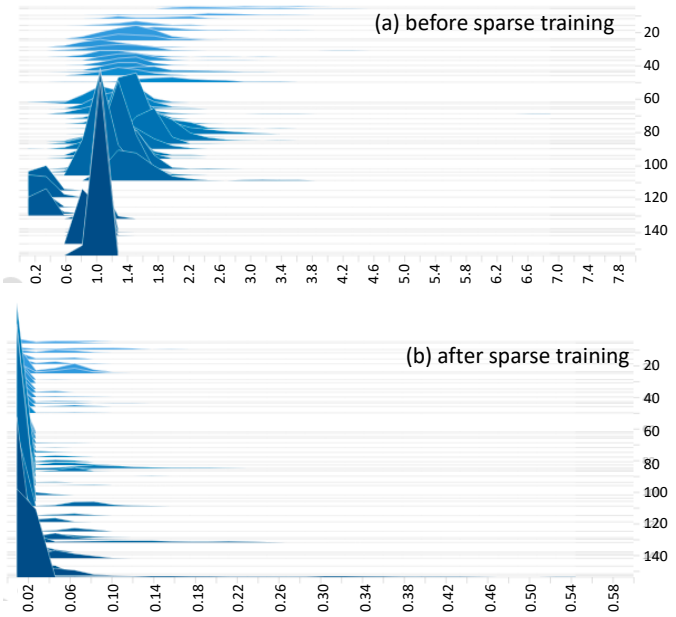

Fig. 4. Gmma weight distribution map of the BN layer

TABLE III
PRUNING RESULTS OF YOLOV4 NETWORK

Step	Precision	Recall	mAP	F1-score	Parameters	BFLOPS
baseline	0.733	0.824	0.817	0.765	64153086	59.919
sparsity training	0.677	0.589	0.624	0.581	64153086	59.919
channel pruning	0.689	0.584	0.62	0.58	1880897	6.685
layer pruning	0.688	0.583	0.619	0.58	1871601	6.663
fine-tuning	0.703	0.796	0.77	0.734	1871601	6.663

curacy detection rates and low power consumption. In general the deployment flow compromises two stages. In the first stage, the weights and/or activations are quantized to the desired bit-width and representation (FP16, INT8). This is done using a heuristic method that benefits from selected pool of images from the training dataset which is so called the golden reference pool. In the second stage, the quantized model is compiled to generate the instruction sequence. In this stage, the model is optimized according to the targeted device.

Three different GPU-based, ASIC-based and FPGA platforms are considered in this study. Their hardware specifications are given in Table. IV, which presents the technical specifications of the embedded edge devices targeted in this work.

In the following subsections, we present the deployment of the trained YOLOv4 models on targeted embedded edge devices: (1) Nvidia Jetson Xavier AGX, (2) Xilinx-AMD Kria VISION AI KV260 and (3) Intel Movidius Myriad X Vision Processing Unit (VPU) integrated in OAK-D camera kit.

A. Deployment targeting Nvidia Jetson Xavier AGX

The trained models are deployed with TensorRT in order to achieve lower latency and higher throughput inference while running on Nvidia platforms. TensorRT is a software develeopment kit (SDK) provided by Nvidia for high performance deep learning inference. It is compatible with most deep-learning frameworks and is used to achieve high performance and platform portability. It compromises an inference optimizer that implements several techniques such as kernal fusion, precision calibration, kernel auto-tuning, dynamic tensor memory and multi-stream execution to optimize the inference of the trained model.

Since *Darknet* framework is not supported by TensorRT, the target models are first converted using Open Neural Network Exchange (ONNX) and then to TensorRT engine with FP32 representation. Next, the target models are quantized to FP16 and INT8 representations. Table. V provides the obtained results in terms of detection performance and inference speed. The table shows a comparison between 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 with slight degradation of mAP for all tested networks. In addition, the quantization leads to better inference rate. For YOLOv4 Tiny network, the inference rate is increased by $\times 1.55$ and $\times 1.67$ times but at the cost of degradation in the detection performance by 2.2 and 2.4 points in mAP for FP16 and INT8 quantization respectively. For YOLOv4 model, the FP16 quantization leads to increase the FPS by $\times 2.93$ and $\times 8.47$ times with a slight degradation in the mAP of 2.9 and 4.1 points for 416×416 and 608×608 input image resolutions respectively. Compared to the results of FP16 quantization, the quantization process for INT8 representation leads to an increase in inference rate by $+6$ and $+4$ FPS but at high cost accuracy degradation of 32.4 and 18.5 point in mAP. It is noted that the quantization of FP16 leads to same mAP value of FP32 but with better inference speed. Note that the FPS are recorded for input HD video with resolution of 1920×1080.

B. Deployment targeting Xilinx-AMD Kria KV260 AI Vision Kit

Vitis AI open source tooset does not support *Darknet* framework. Hence, the trained models are converted to a frozen *TensorFlow* graphs. Note that *MISH* activation function is not supported by the DPU in Xilinx FPGA. So, only models with *LeakyRelu* activation functions are deployed on Kria KV260 AI VIsion Kit. Using VITIS AI toolset, the converted models can be quantized into INT8 representation and then compiled targeting DPUCZDX8G architecture. Indeed, FP16 representation is not supported for Kria KV260 Vision AI kit (Table I). Table. VI shows the obtained results in terms of detection performance and inference speed of the deployed models on Kria KV260 kit. The quantized YOLOv4 Tiny model can achieve the highest inference speed (50 FPS) with a 4.5 points degradation in mAP. The quantized YOLOv4 model achieves the best mAP value (74.9%) on Kria KV260 kit with an inference speed of 11 FPS. The pruned YOLOv4 network can achieve 50 FPS but with significant degradation in mAP (-23.9 points). As a result, YOLOv4 Tiny would be the most convenient choice to run on Kria KV260 kit

TABLE IV
SPECIFICATIONS OF TARGET EDGE EMBEDDED DEVICES

Target Device	Nvidia Jetson Xavier AGX	Kria KV260 Vision AI Kit	Luxonis camera OAK-1-POE
Edge accelerator	512 Core Volta with 64 Tensor Cores	1x DPU configurations B4096 at 300 MHz	Intel Movidius Myriad X VPU
AI Performance (estimated FP16)	11 TFLOPS	FP16 not supported	$\ll$ 1 TFLOPs
AI Performance (estimated INT8)	32 TOPS	1.43 TOPS	INT8 not supported
Max Power consumption	30 W	8 W	4 W
Price	600$	300$	249$

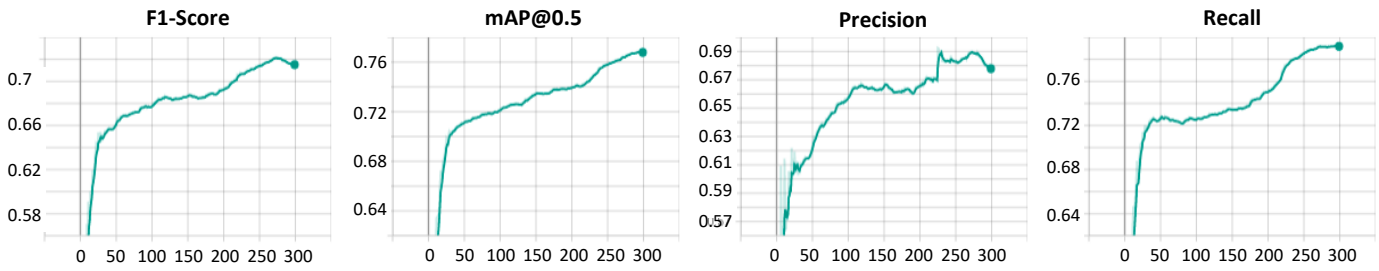

Fig. 5. Obtained performance metrics while fine-tuning the compact model

TABLE V
OBTAINED RESULTS ON JETSON XAVIER AGX

Network	**YOLOv4 Tiny**				**YOLOv4**							
Activation Function	*LeakyRelu*				*MISH*							
Input Resolution	**416×416**				**416×416**				**608×608**			
Model	original	FP32	FP16	INT8	original	FP32	FP16	INT8	original	FP32	FP16	INT8
mAP VOC12	0.783	0.761	0,761	0,759	0,825	0.796	0,796	0,472	0,841	0.800	0.800	0,615
FPS (HD video of resolution 1920×1080)	58	82	90	97	15	19	44	50	9	9	26	30

TABLE VI
OBTAINED RESULTS ON XILINX-AMD KRIA KV260 AI VISION KIT

Network	**YOLOv4 Tiny**			**YOLOv4**			**Pruned YOLOv4**		
Input Resolution	**416×416**			**416×416**			416 × 416		
Model	original	FP32	INT8	original	FP32	INT8	original	FP32	INT8
mAP VOC12	0.783	0.744	0.738	0.829	0.796	0.749	0.77	0.634	0.531
FPS HD video of resolution 960×540	-	-	60	-	-	11	-	-	50
FPS (HD video of resolution 1920×1080)	-	-	15	-	-	11	-	-	15

TABLE VII
OBTAINED RESULTS ON MOVIDIUS MYRIAD X VPU

Network	**YOLOv4 Tiny**			**YOLOv4**		
Input Resolution	416 × 416			416 × 416		
Activation Function	LeakyRelu			MISH		
Model	original	FP32	FP16	original	FP32	FP16
mAP VOC12	0.783	0.7542	0.7541	0.825	0.7856	0.7854
FPS (video of resolution 416 × 416)	-	-	31	-	-	3

as it achieves the highest inference speed and highest mAP value. The FPS values for FP32 representation are not listed in Table VI as the DPU in Kria KV260 kit can run only values in INT8 representation. Note that the FPS values in the table corresponds to processing HD video frames using 4 threads.

C. Deployment targeting Movidius Myriad X VPU

The YOLOv4 and YOLOv4 Tiny models are also deployed on Movidius Myriad X VPU, which is programmable with the Intel distribution of the OpenVINO [40]. The trained models using Darknet framework are first converted to TensorFlow freezed graphs. The converted model is optimized and quantized to FP16 representation using OpenVINO. Indeed, the INT8 representation is not supported for Movidius Myriad X VPU (Table I). Table VII presents the obtained results in terms of detection performance and inference speed. The results show that the FP16 quantized model of YOLOv4 Tiny achieves an inference speed of 31 FPS (2 threads running on 6 SHAVE cores) with a negligible degradation of 0.01% in the mAP (75.41%) when compared to the unquantized converted network (75.42%). The quantized YOLOv4 model achieves a speed rate of 3 FPS (2 threads running on 6 SHAVE cores) when processing a video of resolution 416×416. The degradation of mAP for the quantized YOLOv4 model is 0.02%

D. Analysis

The Jetson Xavier AGX is the most powerful device among the three platforms in terms of detection performance. It achieves the highest inference speed while maintaining acceptable precision values. However, it requires more power budget. It achieves 3.23 FPS/Watt for YOLOv4 Tiny and 1.46 FPS/Watt for YOLOv4.

The KRIA KV260 AI VISION KIT power consumption is 4 times less than the GPU on Jetson Xavier AGX. However, the achieved inference speed is 1.6 times and 4 times less than that achieved using the GPU for YOLOv4 Tiny and YOLOv4 respectively. It is necessary to realize a graph pruning to achieve 15 FPS with only one DPU B4096@300Mhz. While considering

performance per watt criterion only, the KRIA KV260 kit outperforms the other targeted devices when running YOLOv4. It achieves 1.875 FPS/Watt.

The MOVIDIUS MYRIAD X VPU suffers from a lack of computing power, yet it is enough to infer a Yolov4 Tiny at 31 FPS with a power consumption less than 5 Watts with OAK-D camera. The comparison in terms of performance per Watt shows that for YOLOv4 Tiny, the VPU outperforms the other two devices as it achieves 7.75 FPS/Watt.

VI. Conclusion

This paper tackles the topic of marine object detection using deep learning techniques on embedded edge devices. A novel dataset with the widest number of classes and annotations is presented. Several YOLO models with different structure specifications are trained and evaluated considering different parameters such as image resolution and data augmentation. The deployment of the trained models on recent edge devices is considered. Several optimization techniques are applied to enhance the inference speed while maintaining high detection performance. The impact of these techniques in terms of FPS and precision is analyzed while targeting Nvidia Jetson Xavier AGX, AMD-Xilinx Kria KV260 Vision AI Kit, and Movidius Myriad X VPU. The obtained results show promising results. For example, 90 FPS inference speed is achieved on Jetson Xavier AGX with limited degradation of 2.4% in mAP.

References

[1] F. E. Schöller *et al.*, "Assessing deep-learning methods for object detection at sea from LWIR images," *IFAC-PapersOnLine*, vol. 52, no. 21, pp. 64–71, 2019, iFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS.

[2] M. Blanke *et al.*, "Outlook for navigation - comparing human performance with a robotic solution," in *Proceedings of the 1st International Conference on Maritime Autonomous Surface Ships (ICMASS)*. SINTEF, 2019, international Conference on Maritime Autonomous Surface Ships ; Conference date: 08-11-2018 Through 09-11-2018.

[3] D. K. Prasad *et al.*, "Video processing from electro-optical sensors for object detection and tracking in a maritime environment: A survey," *IEEE Trans. on Intelligent Transportation Systems*, vol. 18, no. 8, pp. 1993–2016, 2017.

[4] Z. Bi *et al.*, "A survey of ship target feature extraction based on video surveillance," *Journal of Physics: Conference Series*, vol. 1976, no. 1, p. 012014, jul 2021.

[5] D. K. Prasad *et al.*, "Challenges in video based object detection in maritime scenario using computer vision," *Inter. Journal of Computer and Information Engineering*, vol. 11, no. 1, pp. 31 – 36, 2017.

[6] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *Proc. of the IEEE Computer society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 886–893, 2005.

[8] R. Girshick *et al.*, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587.

[9] R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[10] S. Ren *et al.*, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, C. Cortes *et al.*, Eds., vol. 28. Curran Associates, Inc., 2015.

[11] 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.

[12] 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.

[13] R. Zhang *et al.*, "Survey on deep learning-based marine object detection," *Journal of Advanced Transportation*, vol. 2021, 2021.

[14] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[15] ——, "YOLOv3: An incremental improvement," 2018.

[16] 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

[17] "COCO - common objects in context web site," https://cocodataset.org/, accessed: 2020-06-20.

[18] M. Tan *et al.*, "Efficientdet: Scalable and efficient object detection," in *Proc. of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 781–10 790.

[19] T.-Y. Lin *et al.*, "Focal loss for dense object detection," in *Proc. of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[20] U. Nepal and H. Eslamiat, "Comparing YOLOv3, YOLOv4 and YOLOv5 for autonomous landing spot detection in faulty UAVs," *Sensors*, vol. 22, no. 2, 2022.

[21] M. M. Zhang *et al.*, "Vais: A dataset for recognizing maritime imagery in the visible and infrared spectrums," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2015, pp. 10–16.

[22] E. Zhang *et al.*, "Classification of marine vessels with multi-feature structure fusion," *Applied Sciences*, vol. 9, no. 10, 2019.

[23] A. Khellal *et al.*, "Convolutional neural network based on extreme learning machine for maritime ships recognition in infrared images," *Sensors*, vol. 18, no. 5, 2018.

[24] *Marvel: A Large-Scale Image Dataset for Maritime Vessels*. Asian Conference on Computer Vision (ACCV), 2016.

[25] M. Leclerc *et al.*, "Ship classification using deep learning techniques for maritime target tracking," in *International Conference on Information Fusion (FUSION)*, 2018, pp. 737–744.

[26] "shipspotting website," https:///www.shipspotting.com/, accessed: 2022-07-20.

[27] D. K. Prasad *et al.*, "Object detection in a maritime environment: Performance evaluation of background subtraction methods," *IEEE Trans. on Intelligent Transportation Systems*, vol. 20, no. 5, 2019.

[28] S. Moosbauer *et al.*, "A benchmark for deep learning based object detection in maritime environments," in *IEEE/CVF Conf. on Computer Vision & Pattern Recognition Workshops (CVPRW)*, 2019, pp. 916–925.

[29] J.-H. Kim *et al.*, "Object detection and classification based on YOLO-V5 with improved maritime dataset," *Journal of Marine Science and Engineering*, vol. 10, no. 3, 2022.

[30] Z. Shao *et al.*, "Seaships: A large-scale precisely annotated dataset for ship detection," *IEEE Transactions on Multimedia*, vol. 20, no. 10, pp. 2593–2604, 2018.

[31] B. Bovcon *et al.*, "Mods–a USV-oriented object detection and obstacle segmentation benchmark," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–16, 2021.

[32] M. Kristan *et al.*, "Fast image-based obstacle detection from unmanned surface vehicles," *IEEE Transactions on Cybernetics*, 2015.

[33] Y. Peng *et al.*, "Development of the USV 'jinghai-i' and sea trials in the southern yellow sea," *Ocean Engineering*, vol. 131, pp. 186–196, 2017.

[34] B. Bovcon *et al.*, "The MaSTr1325 dataset for training deep USV obstacle detection models," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 3431–3438.

[35] Y. Zheng and S. Zhang, "Mcships: A large-scale ship dataset for detection and fine-grained categorization in the wild," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2020, pp. 1–6.

[36] B. Iancu *et al.*, "ABOships—an inshore and offshore maritime vessel detection dataset with precise annotations," *Remote Sensing*, vol. 13, no. 5, 2021.

[37] J. Redmon, "Darknet: Open source neural networks in C," http://pjreddie.com/darknet/, accessed: 2022-04-14.

[38] M. Tian *et al.*, "Pruning-based YOLOv4 algorithm for underwater gabage detection," in *Chinese Control Conference (CCC)*, 2021, pp. 4008–4013.

[39] Z. Liu *et al.*, "Learning efficient convolutional networks through network slimming," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2755–2763.

[40] 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.