2022 IEEE International Workshop on Metrology for the Sea; Learning to Measure Sea Health Parameters (MetroSea) | 978-1-6654-9942-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/METROSEA55331.2022.9950745

Deep models optimization on embedded devices to improve the orientation estimation task at sea.

1st Paolo Russo
Department of Computer, Control and Management Engineering "Antonio Ruberti"
Sapienza University of Rome
Rome, Italy
paolo.russo@diag.uniroma1.it

2nd Fabiana Di Ciaccio
Department of Science and Technology
Parthenope University of Naples
Naples, Italy
fabiana.diciaccio@studenti.uniparthenope.it

***Abstract*—The environmental monitoring task has greatly benefited from the improvements achieved in the robotics field. The enhancement of navigation and control algorithms, together with the use of performing, small and low-cost sensors, allows in fact to reduce the implementation costs while improving the system reliability. This is strongly supported by the developments of embedded hardware, smart computing devices able to collect and process data in real-time and in low-resource settings. Following the results obtained by DOES, this work aims at putting another step towards its deployment in live scenarios: we propose a study on the performances of DOES tested on embedded systems, using lighter backbone architectures and model optimization techniques.**

***Index Terms*—Deep Learning, Deep architecture optimization, Environmental monitoring, Embedded systems, Orientation Estimation.**

I. Introduction

In the last decades, particular emphasis has been placed upon the sustainable and effective monitoring of the environment. In this context, research strategies and technological improvements assume an essential role and are greatly benefiting from the enhancements of robotic systems. Autonomous and Remotely Operated Vehicles allow in fact for a complete understanding of the environment, thanks to the different sensors mounted onboard, and can assist or replace the operators depending on the mission objective and risk level as in the case of anti-mine operations or, especially underwater, surveys characterized by long execution times. For this reason, researchers continue to deepen their study towards more advanced solutions with the aim of maximizing the missions productivity whilst keeping low operational costs. In this regards, recent studies and applications proved the robustness of low-cost technologies, confirming their potential in the development of sensing systems able to provide accurate spatial and temporally disperse data. Among the others, the orientation estimation task (at the core of a correct positioning of the vehicles) is now exploited through the Attitude and Heading Reference System (AHRS), which is a cost-effective device integrating the measures of gyroscopes, accelerometers and magnetometers to output the vehicle orientation. The attitude estimation techniques based on low-cost sensors are in continuous development, leveraging on the combined use of inertial and visual data to improve the final system accuracy. In this context, the impressive results obtained by Deep Learning (DL) with its Deep Neural Networks are contributing to enforce these integrated systems, thanks to their great robustness to camera parameters and challenging environments. Moreover, in this era of smart devices, further importance is being given to embedded technologies, electronic or electro-mechanical systems on which microprocessors and microcontrollers are mounted: they can be properly be considered as small computers, as they can perform real-time tasks on the basis of sensors and output actuators [1]. In particular, the former typically examines the behavior of the environment it is placed within and sends the information to an embedded microcontroller system. The actuator then responds to this input performing the required task on its basis and can thus be considered as the output device. can be categorized on the basis of their processing power, cost, architecture and functionality. The small-scale devices, with their 8- or 16- bit architecture and 5V battery power, are successfully employed as the core systems of small-scale robots; in the last years, sensible improvements have been reached on their technology, and these embedded systems are now able to run Deep Learning algorithms in real-time. This work follows the project of DOES [2], a DL based model which aims at improving the attitude estimation at sea and further providing a smart, light-weight and low-cost embedded system to support real-

978-1-6654-9942-2/22/$31.00 ©2022 IEEE

time orientation determination and different monitoring operations of vehicles and robots at sea. In particular, this paper presents: (i) the results of a survey made on the backbone architecture of DOES in view of the recent models developments, searching for enhancements in its structure and weights to improve the inference speed and accuracy; (ii) the study of performance modifications while applying speed-up techniques on the backbone architecture; (iii) the analysis of inference speed on Nvidia embedded devices, with the final aim to deploy DOES on a vehicle in real-time. Section II presents a brief overview of the current literature, with an emphasis on the current DL solutions like DOES and on the state of the art devices capable of DL inference. Section III contains the methodologies at the basis of the model enhancements, and discusses the characteristics of the embedded devices. Section IV shows the experiments and some preliminary results with Section V concluding with some final considerations.

II. Related works

With the aim of providing a supportive visual-based, low-cost technique for attitude estimation, DOES has been developed as a new DL model which outputs the angle of roll and pitch through the processing of the sea horizon view recorded by a low-cost camera. DOES can be combined to a low-cost IMU-based configuration and does not need to take into account camera models or related calibration issues [3].

However, one of the main challenges of DL is the need of huge computation resources: the Graphical Processing Unit (GPU) has been always considered as the best choice by virtue of its high parallelism processing, large integrated memory and bandwidth [4]. The demand for low-consumption, small and inexpensive devices capable of running deep methods drove the development of embedded processing units suitable for mobile robotics and in general small- and medium-scale applications. Some examples of those devices are the Raspberry Pi [5], the Google Coral TPU [6] and the NVIDIA Jetson™ platforms [7].

Researchers in [8] compared the performances of different single-board computers/embedded devices, obtaining that Raspberry Pi generally provides low power and energy saving performance, but the NVIDIA Jetson platforms get higher performance thanks to their higher speed GPUs.

Several studies have been conducted on these platforms; for example, Basulto-Lantsova et al. [9] performed a comparison between the Jetson Nano and TX2 development kits, when implementing the Template Matching method, in order to get an evaluation criterion to select one of them in image processing projects. The Jetson Nano showed two times lower performance, with the TX2

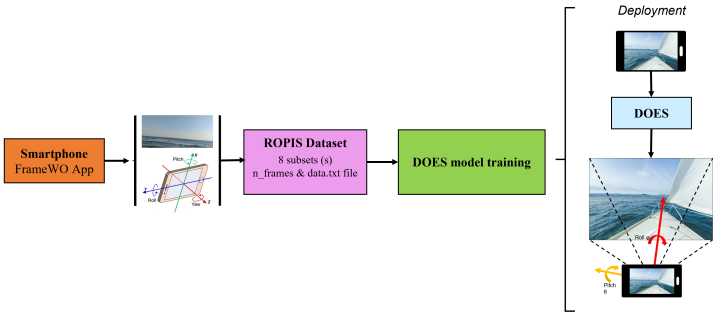

Fig. 1. DOES development workflow [2].

obtaining higher efficiency also on other CNN (Convolutional Neural Network) architectures. The Nvidia Jetson TX2 and the Intel Movidius [10] single- board computers have been tested with some of the most popular DL architectures for object detection (i.e., YOLO, SSD, RCNN, R-FCN and SqueezeNet) using both the CPU and the GPU [11]. Researchers in [12] evaluated the parallel computing performance on Jetson TX1 and Raspberry Pi: the TX1 has been observed to be more effective with a low energy consumption, while the Pi long execution times make it unsuitable for high performance processes.

III. Method

DOES is a deep model for orientation estimation which takes as input an RGB image containing the horizon line at sea and produces as output an estimation of the device roll and pitch angles (Fig. 1).

The main component of DOES is a deep neural network, trained on Pytorch framework, which produces a set of visual features taken as input by two fully connected layers (FC) for the angles regression task. The ResNet18 deep model has been chosen among the others as the default DOES backbone since it produced the best accuracy while keeping a fast inference speed. The backbone has been trained using a standard fine-tuning procedure: the model convolutional kernels were pre-trained on ImageNet dataset while the FC layers have been trained from scratch. Both convolutional and FC layers have been trained using the Adam optimizer [13] and a fixed learning rate set to 0.001 using a standard Mean Square Error Loss (squared L2 norm). The ROPIS dataset [2] has been used for the training: it contains 22173 images of the sea horizon acquired in different locations and environmental conditions.

Starting from the DOES default architecture as baseline model, we performed an optimization for its deployment on the Nvidia Jetson TX1 and Jetson Nano embedded systems. This has the aim of assessing their suitability for being mounted on a robotic vehicle

as operative supporting system for different monitoring operations at sea.

A. Nvidia Jetson embedded systems

DOES has been originally tested on a traditional workstation equipped with an i7 core CPU and a Nvidia Titan X GPU, characterized by plenty of computational power and GPU memory.

The Nvidia Jetson TX1 is an embedded system-on-module built around the NVIDIA Maxwell™ GPU architecture with 256 CUDA cores delivering over 1 TeraFLOPs of performance. The CPU is a Quad-Core ARM Cortex-A57 MPCore, with 4GB 64-bit memory. The Nvidia Jetson Nano is a more recent, entry-level board of the NVIDIA Jetson ecosystem: a small, unexpensive and powerful single-board with the same characteristics of the TX1 except for the GPU, which has instead 128 CUDA cores. For both the hardwares a development kit supported by NVIDIA gives support for many common APIs and Deep Learning frameworks and runtime (e.g. Pytorch, Tensorflow, ONNX).

B. Fast-inference backbones

As previously mentioned, DOES model is composed of a pre-trained CNN backbone and two additional FC layers to output the roll and pitch estimates. The ResNet18 model has been originally chosen as the default DOES backbone by virtue of its good performances over a lower number of trainable parameters ($11M$). However, to further improve both the inference speed and the power complexity requirements, two more lightweight models have been tested: the MobileNetV3 [14] and the EfficientNet b0 [15].

ResNet [16] is a family of deep models based on the *residual* architecture. It is made of a series of residual blocks in which the feature maps calculated by the convolutional layers are added to the input, so that each residual block calculates an *update* (hence residual) of the input feature maps. This approach makes the network resilient to the vanish gradient problem [17], improving both the convergence speed and the final accuracy. Moreover, all the ResNet models avoid the use of the FC layers after the convolutional blocks, reducing the total number of trainable parameters and thus lessening the overfitting effect on training data. MobileNets introduced the *depthwise* separable convolutions, which can effectively factorize traditional convolutions by separating spatial filtering from the feature generation mechanism. Depthwise separable convolutions are defined by two different layers: a light weight depthwise convolution for spatial filtering and a heavier 1x1 pointwise convolutions for feature generation. The MobileNetV3, in particular, uses a combination of these layers with those introduced by the MobileNetV2 [18] and the MnasNet [19] as building blocks to improve the effectiveness of the model. More in detail, the MobileNetV2 uses linear bottleneck and inverted residual structure to define the structure with a 1x1 expansion convolution followed by depthwise convolutions and a 1x1 projection layer. Moreover, the input and output are connected with a residual connection if and only if they have the same number of channels. This structure maintains a compact representation at the input and the output while expanding to a higher-dimensional feature space internally to increase the expressiveness of nonlinear perchannel transformations. The MnasNet takes inspiration by the previous model and further introduces lightweight attention modules based on squeeze and excitation into the bottleneck structure. In the expansion, this module follows the depthwise filters to apply the attention on the largest representation.

The EfficientNet family achieves great performances by uniformly shaping depth, width, and resolution while scaling down the model. This allows to balance all the dimensions of the network with respect to the available resources, thus effectively improving the overall performance. We tested the b0 version as it is the smallest model w.r.t. the number of parameters.

The main building block of this architecture is the inverted bottleneck MBConv, which was first introduced in the MobileNetV2 as it noticeable increases the FLOPS (floating point operations per second) budget. Moreover, EfficientNet is much smaller than the other models: there is a total of 8 models between B0 and B7, in which the number of parameters slightly increases with the model number but does not affects the performances. For example, the ResNet50 model with its 23.5M parameters still underperforms the smallest EfficientNet (the EffecientNet-B0, which will be used in this work), which only has 5.3M parameters in total, when testing on standard benchmark datasets like ImageNet [20].

In the experiments presented in this work, all the networks have been fine-tuned on the proposed ROPIS dataset starting from the ImageNet [21] pre-trained weights. The ResNet18 has been chosen among the others as the default DOES backbone since it produced the best accuracy while keeping at the same time a fast inference speed. Fig. 1 reports the DOES network with the default ResNet18 backbone.

C. Model modification and conversion

The *Open Neural Network Exchange* (ONNX) is an open format created for deep models interoperability. Most of the available Deep Learning frameworks (e.g. Pytorch, Tensorflow, Mxnet) allow to export to the ONNX format, which performs a series of neural network graph optimization in order to get a compact, fast model representation which can be run on heterogeneous devices. Among

the ONNX optimizations, the fusion of convolutional, activation and batch normalization layers into a single integrated layer is the one which can provide the major speed-up in the inference phase. As several ONNX runtime backends are available for the inference execution, we have chosen the *CUDA Provider* and the *CPU Provider* to be run the GPU and the CPU respectively.

The *Pruning* technique is a simple approach for model parameters reduction. It consists in removing the convolutional and FC weights which show the minimum output variance. This usually has a small impact on the overall network performances, but produces a sparse matrix which further requires an optimized sparse calculation framework to fasten the inference. Among all the possible methods developed in the scientific literature, we exploited a simple unstructured Pruning technique [22], with a Pruning factor of 0.5 for a 50% reduction of the number of parameters.

Model Compression tries to reduce the model number of parameters by changing the network shape and structure. One of the most promising techniques is based on the low rank tensor approximation, consisting in the layer weights decomposition through a Tucker function and a further substitution with a block of smaller layers obtained by minimizing a Frobenius norm [23] [24].

IV. Experiments and results

Table I reports the performances of the ResNet18, MobileNetV3 end EfficientNet b0 backbones of DOES both with Pytorch and ONNX frameworks. The three networks produce good accuracy estimations, with a Mean Absolute Error (MAE) around 2.0; the ResNet18 performs slightly better than the others, with a difference in the MAE of $0.1 - 0.2$ points.

The inference speed is measured as processed frames per second (fps), and does not consider the data loading phase. Both the error and the speed results are the mean values calculated over ten runs. The ResNet18 obtains the best inference speed on the Nvidia GPU workstation, with a fps that is 2.5x and 3.3x faster than the MobileNet and the EfficientNet respectively. This speeding up could have benefited from the small number of convolutional layers and the high number of convolutional features per layer, which positively impact on the parallelism. Similar considerations can be made on the embedded platforms (for brevity, we here report the Jetson TX1 results), where the default ResNet18 is sensibly faster both on the GPU and the CPU, with the MobileNet and EfficientNet unable to reach one frame per second on the ARM CPU. However, a different scenario is obtained when taking into account the model conversion to ONNX format. In this optimized representation, the models get an average speed-up of 100x-1000x which enables fast inference for all the considered hardwares. For this reason we highly recommended the ONNX conversion before the model deployment on embedded systems.

TABLE I
DOES Results on the tested backbones, in terms of Mean Absolute Error(MAE) on roll and pitch angles and Frames Per Second (FPS). (O) indicates the run on the ONNX framework.

Backbone	MAE		Workstation	Jetson TX1	
	Roll	Pitch	GPU fps	GPU fps	CPU fps
Resnet18	**1.73**	**1.89**	**184.77**	**29.95**	**3.01**
MobileNet	1.93	2.01	87.71	13.01	0.30
EfficientNet	1.80	2.21	58.51	9.02	0.09
(O) Resnet18	**1.73**	**1.89**	**22727.26**	2881.84	334.56
(O) MobileNet	1.93	2.01	22026.43	3436.42	**1626.01**
(O) EfficientNet	1.80	2.21	20000.19	1904.76	431.03

Table II shows the results of the different modifications applied to the backbone models after ONNX conversion. The Pruning has a negative impact on the evaluation metrics (especially on the pitch angle), without introducing any improvement in the inference phase. This is probably due to the inefficient sparse tensor calculations of both the deep learning frameworks and the current hardware, which are developed around the concept of dense matrix operations. The Compression approach is able to significantly reduce the number of matrices operations by splitting the original layers in an increased number of smaller layers. This technique produces a remarkable speed improvement when applied on the ResNet/CPU combination, while it fails to converge on the MobileNet. In general this modification comes with a small increase of error. Unfortunately, the EfficientNet powerful pretrained features do not correspond to a consistent decrease of the MAE produced by DOES, and its smart structure is not reflected into faster estimations: for this reason, the EfficientNet b0 is not the suggested choice for the real-time orientation estimation of DOES. Finally, all the tested models require a memory allocation ranging between 2GB and 3GB, which makes them able to run on the considered 4GB embedded systems.

V. Conclusions

This work presents the preliminary results of a number of experiments made to enhance DOES in its deployment on embedded devices, in particular those of the NVIDIA Jetson series. DOES has in fact been developed as a supportive low-cost technology to improve the vehicle attitude estimation and the related monitoring operations at sea, thus requiring affordable and robust embedded platforms. In the light of the obtained results we strongly recommend the use

TABLE II
DOES RESULTS ON THE UNMODIFIED BACKBONES, REPRESENTED BY THEIR FIRST LETTER NAME (R,M,E), COMPARED WITH THE CORRESPONDING MODEL AFTER PRUNING OR COMPRESSION ENHANCEMENT. ALL THE RUNS ARE PERFORMED ON THE ONNX FRAMEWORK.

Method	MAE		Workstation	Jetson TX1	
	Roll	Pitch	GPU fps	GPU fps	CPU fps
(R) Original	**1.73**	**1.89**	**22727.26**	2881.84	334.56
(R) Pruning	2.10	7.16	22727.18	2881.84	332.22
(R) Compression	2.13	2.61	22222.24	3039.51	698.81
(M) Original	1.93	2.01	22026.43	3436.42	**1626.01**
(M) Pruning	4.94	5.34	22727.27	3436.42	1579.77
(M) Compression	-	-	-	-	-
(E) Original	1.80	2.21	20000.19	1904.76	431.03
(E) Pruning	3.19	9.80	20120.72	1904.76	431.03
(E) Compression	3.11	3.68	19417.47	1953.12	536.19

of the ONNX format for any of the proposed backbones; in this context, the ResNet18 produces the lowest estimation error while the MobileNetV3 shows the fastest inference speed. The latter has a high-speed estimation throughput even on the low powered ARM CPU of the Jetson modules, making it a valuable candidate also for other embedded devices (e.g., smartphones, Raspberry systems).

At the same time, the other model enhancements produce mixed results, with the compression technique being able to improve the inference speed only on sequential computation hardware such as the Jetson TX1 and Nano ARM CPU, while failing to produce a stable model on the MobileNet. Finally, the Pruning technique, despite being easy to use, does not provide any speed improvement due to the lack of sparse tensors calculations support. Future works will focus on the development of new model reduction techniques compatible with the MobileNet and ResNet architectures. The quantization approach will also be deepened as it could provide similar estimation accuracy at a fraction of the memory consumption: this will give the possibility to simultaneously run DOES with different cameras on the same device, thus improving the reliability of the estimations. Moreover, the high framerate obtained in the experiments unlocks the future development of deeper and more complex models, with a further increase in the estimation accuracy. These modifications will positively impact the robustness of DOES towards its deployment as a low-cost supportive technology for an advanced environmental monitoring system.

REFERENCES

[1] N. Dey and A. Mukherjee, *Embedded systems and robotics with open source tools*. CRC Press, 2018.

[2] F. Di Ciaccio, P. Russo, and S. Troisi, "Does: A deep learning-based approach to estimate roll and pitch at sea," *IEEE Access*, vol. 10, pp. 29 307–29 321, 2022.

[3] S. Poddar, R. Kottath, and V. Karar, "Evolution of visual odometry techniques," *arXiv preprint arXiv:1804.11142*, 2018.

[4] M. Afif, R. Ayachi, Y. Said, and M. Atri, "An evaluation of efficientdet for object detection used for indoor robots assistance navigation," *Journal of Real-Time Image Processing*, vol. 19, no. 3, pp. 651–661, 2022.

[5] E. Upton and G. Halfacree, *Raspberry Pi user guide*. John Wiley & Sons, 2014.

[6] S. Cass, "Taking ai to the edge: Google's tpu now comes in a maker-friendly package," *IEEE Spectrum*, vol. 56, no. 5, pp. 16–17, 2019.

[7] H. Halawa, H. A. Abdelhafez, A. Boktor, and M. Ripeanu, "Nvidia jetson platform characterization," in *European Conference on Parallel Processing*. Springer, 2017, pp. 92–105.

[8] A. A. Süzen, B. Duman, and B. Şen, "Benchmark analysis of jetson tx2, jetson nano and raspberry pi using deep-cnn," in *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*. IEEE, 2020, pp. 1–5.

[9] A. Basulto-Lantsova, J. A. Padilla-Medina, F. J. Perez-Pinal, and A. I. Barranco-Gutierrez, "Performance comparative of opencv template matching method on jetson tx2 and jetson nano developer kits," in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2020, pp. 0812–0816.

[10] G. Dinelli, G. Meoni, E. Rapuano, G. Benelli, and L. Fanucci, "An fpga-based hardware accelerator for cnns using on-chip memories only: Design and benchmarking with intel movidius neural compute stick," *International Journal of Reconfigurable Computing*, vol. 2019, 2019.

[11] C. E. Kim, M. M. D. Oghaz, J. Fajtl, V. Argyriou, and P. Remagnino, "A comparison of embedded deep learning methods for person detection," *arXiv preprint arXiv:1812.03451*, 2018.

[12] P. M. Pereira, P. Domingues, N. M. Rodrigues, G. Falcao, and S. M. Faria, "Assessing the performance and energy usage of multi-cpus, multi-core and many-core systems: the mmp image encoder case study," *International Journal of Distributed and Parallel Systems*, vol. 7, no. 5, pp. 1–20, 2016.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

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

[15] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[17] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," *Advances in neural information processing systems*, vol. 29, pp. 550–558, 2016.

[18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[19] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile,"

in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2820–2828.

[20] H. Alhichri, A. S. Alswayed, Y. Bazi, N. Ammour, and N. A. Alajlan, "Classification of remote sensing images using efficientnet-b3 cnn model with attention," *IEEE access*, vol. 9, pp. 14 078–14 094, 2021.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[22] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems*, vol. 28, 2015.

[23] L. R. Tucker, "Implications of factor analysis of three-way matrices for measurement of change," *Problems in measuring change*, vol. 15, no. 122-137, p. 3, 1963.

[24] J. Gusak, M. Kholiavchenko, E. Ponomarev, L. Markeeva, P. Blagoveschensky, A. Cichocki, and I. Oseledets, "Automated multi-stage compression of neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.