NERONE: the Fast Way to Efficiently Execute Your Deep Learning Algorithm at the Edge

Raffaele Berzoini‡, Eleonora D'Arnese‡, *Member, IEEE,* Davide Conficconi, *Member, IEEE,* and Marco D. Santambrogio, *Senior Member, IEEE*

***Abstract*— Semantic segmentation and classification are pivotal in many clinical applications, such as radiation dose quantification and surgery planning. While manually labeling images is highly time-consuming, the advent of Deep Learning (DL) has introduced a valuable alternative. Nowadays, DL models inference is run on Graphics Processing Units (GPUs), which are power-hungry devices, and, therefore, are not the most suited solution in constrained environments where Field Programmable Gate Arrays (FPGAs) become an appealing alternative given their remarkable performance per watt ratio. Unfortunately, FPGAs are hard to use for non-experts, and the creation of tools to open their employment to the computer vision community is still limited. For these reasons, we propose NERONE, which allows end users to seamlessly benefit from FPGA acceleration and energy efficiency without modifying their DL development flows. To prove the capability of NERONE to cover different network architectures, we have developed four models, one for each of the chosen datasets (three for segmentation and one for classification), and we deployed them, thanks to NERONE, on three different embedded FPGA-powered boards achieving top average energy efficiency improvements of $3.4\times$ and $1.9\times$ against a mobile and a datacenter GPU devices, respectively.**

***Index Terms*— Deep Learning, Edge Acceleration, Medical Image Analysis, Energy Efficiency, FPGA.**

I. Introduction

The advent and the spreading of imaging techniques such as Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) allow the analysis of critical anatomical structures [1], [2]. Their employment assures a non-invasive study of internal structures for both screening and diagnostic purposes. Unfortunately, the number of images daily generated and the scarcity of expert radiologists produce a considerable workload causing delays in the intervention [3]. In this context, semantic segmentation and classification play a fundamental role in identifying organs, lesions, and regions of interest, pivotal in many downstream applications such as tumor detection, dose definition, and surgery planning. Precisely, semantic segmentation is a pixel level classification that consists in assigning at each pixel within an image a label representing a class [4], [5]. Such a task is particularly time-consuming if done by hand; therefore, its automation is essential to make it faster, efficient, and standard, allowing the physicians to carry out the diagnosis and the prognosis [6].

Currently, the application of Deep Learning (DL), and in particular Fully Convolutional Networks (FCNs), has delivered state-of-the-art results in segmentation and classification in fields ranging from autonomous driving and robotic navigation to medical practice [7], [8]. Indeed, FCNs predict a value indicating for each pixel or image its class of belonging, providing a fast and easy way to interpret information regarding the observed scene. Developing effective semantic segmentation and classification workflows based on FCNs for medical image segmentation/classification requires vast and properly annotated datasets, which are expensive and difficult to obtain. Moreover, training these models requires computational and energetic resources that are not always available in the medical field [9]. While dataset dimensions problems can be faced by patching images, data augmentation strategies, and generated data [10], [11], the power and computational load needed to perform the segmentation task are still an open challenge.

Currently, Graphics Processing Units (GPUs) are the de facto gold standard for both the training and the inference stage of FCNs. Indeed, not only they deliver notable performance and throughput, but their widespread adoption is also driven by the numerous libraries that support and facilitate making the most out of the GPUs capabilities without knowing parallel programming (e.g., TensorFlow, PyTorch). On the other hand, GPUs are power-hungry devices limiting their adoption in restraint scenarios such as in the surgery room. Indeed, during a computer-aided intervention, we want to perform real-time analyses while limiting power consumption since it is mainly provided to the surgical and imaging pieces of machinery [12]. Therefore, in the last years, new approaches have been explored by companies and research intuitions.

Motivation and Related Work. To offer an alternative to power-hungry GPUs for the inference phase of FCNs, Field Programmable Gate Arrays (FPGAs) are appealing devices for their fine-grained computation capabilities at a reasonable energy budget. Therefore, thanks to the possibility of delivering considerable performance while limiting power consumption, accelerators are being studied as valuable alternatives to GPUs for inference, especially in edge scenarios [12]–[14]. Although there is a massive amount of research to develop fast and energy-efficient custom accelerators in the

Manuscript received: July 31 2022; revised May 20, 2023; Accepted: July 9 2023. (Corresponding Author: Eleonora D'Arnese)
‡equal contributions; R. Berzoini, E. D'Arnese, D. Conficconi, and M. D. Santambrogio are with the Department of Electronic, Information and Bioengineering, Politecnico di Milano, Milano, IT, 20133. E-mail: raffaele.berzoini@mail.polimi.it E-mail: {eleonora.darnese, davide.conficconi, marco.santambrogio}@polimi.it

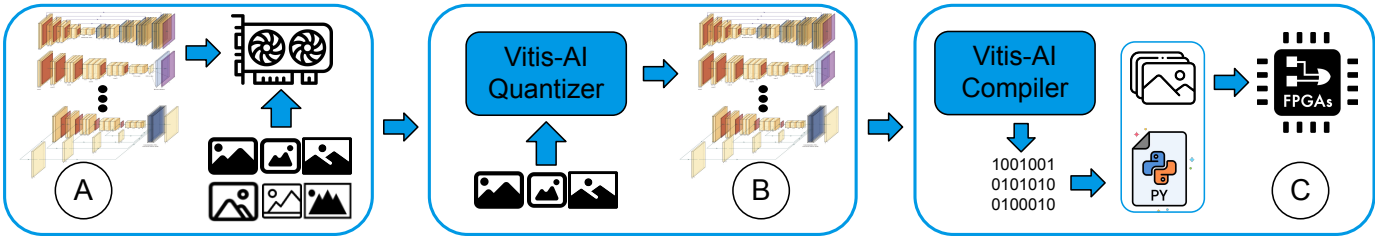

Fig. 1: High-level view of the NERONE framework. It consists of models training on GPU (A). Models quantization with the Vitis-AI quantizer, which makes use of a subset of the original dataset (B). In the end, the Vitis-AI compiler generates the DPU binary instructions based on the quantized network and the target board DPU architecture (C).

edge world, designing them from scratch could require months for a non-expert hardware developer [15], such as a computer vision expert. Moreover, the majority of the available solutions present accelerators implementing a single network structure and allowing its execution on a single or highly restricted number of devices [13]. Additionally, FCNs, and in general, DL algorithms, evolve extremely fast, making those accelerators obsolete fast as well. Therefore, an accelerated and efficient version is required along a high-speed design cycle to keep the evolution pace. While GPU-enhanced frameworks widely support these computations, FPGA-based ones still present a barrier to computer vision scientists, and although some attempts to lower this barrier have been made, like AMD-Xilinx Vitis-AI [16], they still require some hardware knowledge and basic tools understanding. Based on these considerations, creating automation toolchains that open the employment of FPGA-based DL models will pave the way to broader adoption of such solutions by DL experts and also in the clinical practice exploiting FCNs.

Proposed Solution. For these reasons, we propose NERONE, which, building on top of Vitis-AI [16], exposes a framework that minimizes the required knowledge and efforts while granting the highest energy efficiency at the edge with a reasonable deployment time. Therefore, NERONE poses itself as the tool that allows deep learning and computer vision developers to seamlessly benefit from FPGA acceleration and energy efficiency without modifying their development flows. Indeed, they create and train their model as usual, and NERONE hides the complexity of FPGA-based acceleration with a few Python lines of code. Indeed, NERONE, given a dataset, a trained model, and the platform to be used, automatically deploys the solution for inference.

Our main contributions are:

- A comprehensive framework to deploy DL models on FPGAs to achieve a fast, accurate, and efficient inference to meet the requirements of constrained scenarios such as the medical one (Section II).
- An open-source[1] methodological workflow to deploy, on a user-oriented interface, the FPGA-based models with real-time visualization of network outputs (Section II).
- A systematic design and results analysis to demonstrate the framework's generalizability to a wide range of datasets, models, and tasks (Section III, IV, and V).

[1]https://github.com/necst/NERONE

To prove the generalization of NERONE, we selected four open-source medical datasets (three for semantic segmentation and one for classification) and deployed a different network architecture for each of them. Additionally, we run them on three different embedded FPGA-powered boards. From this experimental evaluation, we achieved top average energy efficiency improvements of $3.4\times$ and $1.9\times$ against a mobile and a datacenter GPU device, respectively.

II. Proposed Approach

This Section describes the NERONE workflow that allows deploying the inference phase of DL models to Field Programmable Gate Array-based (FPGA) platforms. Moreover, thanks to NERONE, it is possible to obtain an efficient and accurate inference on a generic dataset employing a generic Deep Neural Network (DNN) applicable in scenarios where power consumption is a limiting factor. Figure 1 illustrates the steps to obtain a quantized and compiled model executable on an FPGA-based Deep Learning Processor Unit (DPU) [17]. Moreover, thanks to the adaptability of the quantization and compilation, NERONE enables deploying a pretrained model with a small set of images on edge, thus skipping the first steps of the presented work with a small set of input images.

A. Training and Model Preparation

The primary purpose of NERONE is to optimize a DNN to reduce memory bandwidth and power consumption and make it executable on an edge device powered with an FPGA. NERONE does not have particular limitations or actions to be followed during model training. The only constraint is on the version of the framework used for GPU model definition and training that has to be compatible with the Vitis-AI [16] version adopted during quantization and compilation. Since the quantization phase alters the model weights, quantizing a trained model only once it has reached a plateau in its loss function to obtain more stable and predictable performance during and after quantization is also convenient.

B. Model Quantization

Currently, model training on GPU is generally performed with at least 32-bit floating-point (FP32) weights. When the working scenario allows easy use of a GPU, the FP32 weights are often kept at inference time, but this requires a complete working station to support the GPU computations. Conversely,

as shown in literature [13], [18], when the working scenario requires a memory footprint reduction, better energy efficiency, and less voluminous device, models with 8-bit integer (INT8) weights drastically reduce the required memory and power consumption with a negligible accuracy loss. For these reasons, within NERONE, we exploit the Vitis-AI quantizer to convert the FP32 weights of pretrained models into INT8 while maintaining the overall accuracy of the GPU-based models. Given the INT8-weights model, the edge device execution engine, namely the DPU, can perform more energy-efficient computations with a reduced memory demand. Additionally, the Vitis-AI quantizer optimizes the INT8 model by removing inference-useless layers, such as the dropout layers, and folding some layers into preceding ones to reduce the overall number of computations necessary to provide network outputs.

The Vitis-AI quantizer offers three quantization procedures: *Post Training Quantization (PTQ)*, *Fast Finetuning Quantization (FFQ)*, and *Quantization Aware Training (QAT)*. PTQ and FFQ apply quantization using a small set of unlabelled images, namely the calibration dataset, even though PTQ is faster since it converts all the model weights simultaneously. Adopting the FFQ method based on the AdaQuant algorithm, which performs quantization layer by layer, is helpful when facing accuracy losses. The third and slowest method is QAT, which differs considerably from the previous ones. It rewrites the floating-point model converting it into a quantized one before starting training. Unlike PTQ and FFQ, QAT requires the whole dataset (input images and labeled outputs) and longer and more hardware-demanding computations. Within NERONE, we could use PTQ with no global performance loss for all models and datasets. As previously stated, PTQ requires a small calibration dataset; generally, between 100 and 1000 images are enough to guide the quantization. Such a calibration dataset is crucial since the Vitis-AI quantizer performs not only a simple conversion of the weights but also adjusts their values to minimize the output differences between INT8 and FP32 models. Although there are no specific requirements for the dataset size, from our experimental evaluation, having at least a few dozen images for each class is recommended. Indeed, in our case, we consider 500 unlabelled images to convert the FP32 weights into INT8 ones. Finally, the calibration dataset usually reflects the exact distribution of the training data providing good performance; however, sometimes, it may be required to adjust the distribution manually (i.e., increase or decrease the number of images containing a particular class) to obtain homogeneous performance among the classes of the dataset. Therefore, the calibration dataset's distribution and dimension can be considered an additional set of hyperparameters that the user can tune.

C. Model Compilation and Deployment to the Edge

Once we obtain a quantized model with negligible accuracy reduction, we compile, deploy, and evaluate the INT8 model on three target FPGAs from AMD-Xilinx, namely, an Avnet Ultra96-V2, a ZCU104, and a Kria KV260. Before deployment and evaluation, we must generate the binary instruction for our target programmable soft engines, which are the Deep Learning Processor Units (DPUs) of the target FPGAs. Figure 2 reports a system view of the ZCU104 device as an example. It comprises a System-on-a-Chip (SoC) with the main memory off-chip. All three considered SoC have an ARM-based CPU and an FPGA, on which our target DPUs are implemented. The default DPU version for SoC-based devices employs a quantization approach based on INT8 data types for CNNs and has a parallelism degree parameters to regulate the Peak Operations Per Second (POPS). The Ultra96-V2 achieves 2304 POPS, while the Kria KV260 and the ZCU104 achieve 4096 POPS. Moreover, the latter implements a dual-core DPU.

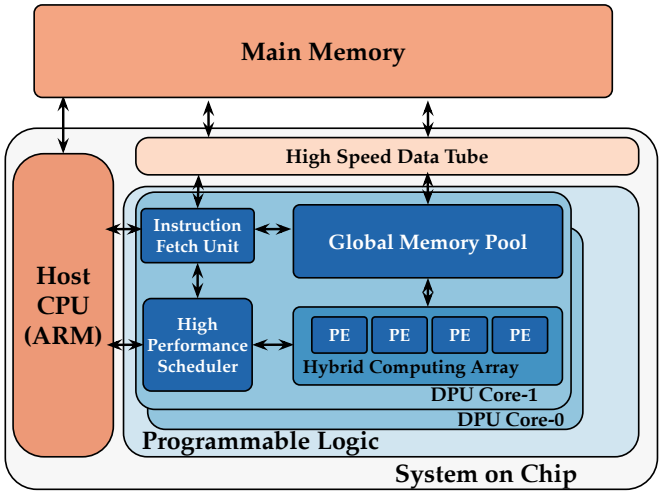

Fig. 2: System view of an example DPU dual-core and its internal architecture and system-wise components

The binary instructions are stored into an *xmodel* file generated using the Vitis-AI compiler (VAI_C). The compiler's framework can parse the quantized network's topology and build an internal computation graph. VAI_C performs additional optimizations, such as scheduling instructions efficiently by exploiting data reuse and parallelism. Finally, the back-end generates the compiled *xmodel* file based on the target DPU microarchitecture. Since the ZCU104 and the Kria KV260 share the same DPU architecture, the *xmodel* file is compatible with the two boards without the need to be recompiled. On the other hand, for the Ultra96, a different *xmodel* file has to be generated. From the user perspective, the whole process requires the execution of two scripts, one for the flow from the quantization to the deployment and one that takes care of the execution of the actual inference. Thanks to this automation, if the end-users need to fine-tune the model after initial deployment, they can do it as usual on CPU/GPU, and thanks to NERONE, a redeployment requires less than 20 minutes, far lower than a hardware redesign cycle.

We deploy our models to the target boards with the binary instruction files. We employed PYNQ-DPU[2], a software layer that runs on our SoCs and allows us to submit it asynchronously and collect jobs to/from the accelerator and take advantage, where present, of the multi-core architecture. PYNQ-DPU offers Python APIs and a user-oriented interface to deploy the application and visualize directly from the board

[2]https://github.com/Xilinx/DPU-PYNQ

the DNN outputs. PYNQ-DPU also automatically performs the preprocessing steps necessary to feed the models with the input images. Indeed, as we quantize our models from FP32 to INT8 weights, they no longer accept the original input images datatypes because they are incompatible with the new network structure and weights. Therefore, the xmodel file stores a set of values with which images has to be scaled. PYNQ-DPU automatically performs this step allowing a more accessible and intuitive user experience. The only drawback of using PYNQ-DPU is that it requires a portion of the available resources, leading to a performance reduction of the quantized models. However, PYNQ-DPU is ideal for prototyping and reaching a more comprehensive range of final users.

III. Experimental Setup

As previously introduced, we have designed four different networks, one per dataset, and trained them on GPU in a classical fashion. For the network structures definition, we exploited TensorFlow 2.3.0 and trained them on an Ubuntu 20.04 Oracle virtual machine with a six-core Intel Xeon Platinum 8167M (Skylake) CPU with 90 GB of memory and a 16 GB NVIDIA Tesla V100. Such a setup was used both for training and testing the baseline configurations; additionally, we also used an Ubuntu 18.04 laptop with an Intel Core i7-10750H and an NVIDIA GeForce RTX 2060 Mobile for further comparison. All the trained models are considered as baselines and the starting point for developing the FPGA-based solutions. Concerning the FPGA side, we exploit Vitis AI 1.4.1, PYNQ-DPU 1.4.0 and the default Deep Learning Processor Units (DPUs), as Section II-C described. Finally, as evaluation metrics, we employ the throughput and Energy Efficiency (EE) for all the proposed models, while the Dice Similarity Coefficient (DSC) for the segmentation-based tasks and the accuracy for the classification one.

1) Throughput and Efficiency: For evaluating NERONE, we employ the Frames Per Second (FPS) as a proxy of the throughput, computing how many images are processed by the selected model in one second. Additionally, we measure the DC power consumption in Watt during the inference phase with the Voltcraft 4000 energy logger for the FPGAs and the *nvidia-smi* command for the RTX2060 Mobile and the TESLA V100. Finally, since throughput and power consumption are dependent quantities, we evaluate the models' efficiency in terms of energy efficiency as:

$$EE = \frac{FPS}{Watt} = \frac{FRAMES}{Joule} \tag{1}$$

2) Performance Metrics: Given the different nature of the two tasks selected in the validation of NERONE, we select two different evaluation metrics accordingly. For the semantic segmentation, we select the widely employed DSC weighted on the class representation in the dataset, defined as:

$$DSC = \sum_{i}^{C} w_i \frac{2\,|P_i \cap G_i|}{|P_i| + |G_i|} \tag{2}$$

where P is the predicted segmentation, G is the ground truth label, and w is the frequency with which the i^{th} class C appears in the dataset. On the other hand, for the multi-class classification tasks, we employ the overall Accuracy.

A. Datasets Description and Pre-Processing

To train and evaluate the performance of NERONE we have employed four open-source datasets representing different anatomies and imaging techniques.

1) Segmentation: For the first segmentation task, we have selected the CT-ORG dataset [19], [20] that provides 140 Computed Tomography (CT) volumes in NIfTI where the lungs, the liver, the bones, the bladder, and the kidneys are segmented and provided as ground truth. We downsized the input images from 512×512 to 256×256, we resampled by cubic interpolation the axial voxel size to $0.557mm\times0.557mm$, and we performed a 3D histogram analysis to make the input images contrast uniform thanks to a linear transformation with input interval [-500, 1000] and output interval [-1, 1]. The second segmentation dataset is BraTS [21]–[23] which reports 369 multi-modal Magnetic Resonance (MR) images and the corresponding manual segmentation of enhancing tumor, the peritumoral edema, and the necrotic and non-enhancing tumor core. We kept the original input dimensions of 240×240. The last segmentation dataset is the Prostate-3T from the NCI-ISBI 2013 Challenge [24] composed of 30 prostate transversal T2-weighted MR image volumes along with the segmentation of the central gland and the peripheral zone obtained with MeVisLab. We reduced the input image dimensions from 320×320 to 256×256, we resampled by cubic interpolation the axial voxel size to $0.625mm\times0.625mm$, and we adjusted the contrast of the images by saturating the upper and lower 1% of the pixels. We rescaled the input images in the [-1, 1] interval for all the segmentation datasets to match the network requirements and the subsequent quantization phase.

2) Classification: We have employed the BCCD dataset [25], a small dataset for blood cell classification which includes 9957 augmented blood cell images from the 410 original ones. The dataset contains approximately 2500 images for each of the four classes: eosinophil, lymphocyte, monocyte, and neutrophil. For this dataset, we kept the starting dimension of 240×320 and the three color channels.

B. Models Employed

To validate NERONE, we employed a different network architecture for each of the four datasets to further prove the adaptability to different models. We employed a 2D FCN based on the U-Net [26] architecture for the CT-ORG dataset [19], [20], while for the Prostate-3T dataset [24], we chose a network based on the V-Net [27] architecture. These two architectures, i.e., U-Net and V-Net, were chosen given their wide employment in biomedical segmentation tasks [28]. For the BraTS dataset [21]–[23], we exploited an FCN decoder attached to a MobileNetV2 [29] used as the encoder path of the architecture. The choice of MobileNetV2 relies on the possibility of testing different parameters and configurations. Moreover, like U-Net and V-Net, it is also employed in different scenarios, including medical ones [30]. Finally, for the classification task on the BCCD dataset [25], we

implemented a 2D CNN [31] based on the ResNet architecture [32]. This architecture is among the most used and efficient for classification tasks making it a good fit for NERONE's evaluation. In addition to supporting almost all architectures available in the state-of-the-art, NERONE has no limitations on model size and can quantize and compile models of any dimension. The only constraints come from the physical device the user chooses as the model deployment device since the available memory varies from one another. For example, on the ZCU104, which has 4096 MB of available memory, the user can accommodate models with up to 198 million parameters.

1) U-Net: comprises four stacks in the encoder path, four in the decoder one, and the stack connecting the two paths. In the encoder path, each stack is made by two *Conv2D-batch normalization-ReLU activation* (CBA) blocks followed by a max pooling layer and a 5% dropout layer. On the other hand, each stack of the decoder path is made by a 2D transpose convolution, a concatenation layer, a 5% dropout layer, and two CBA blocks. In the encoder path, the first stack has eight filters for the 2D convolution, and the number of filters doubles at each stack. In the decoder path, the number of filters halves at each stack. The last layer adopts six 3×3 convolutional filters and a softmax activation function to generate the six probability maps, one for each organ and the background. The final outputs of the U-Net have been obtained thanks to the argmax function applied to the $256 \times 256 \times 6$ output stack.

2) MobileNetV2: is used as the encoder of a more extensive encoder-decoder architecture. The encoder based on the MobileNetV2 [29] structure has a width multiplier of 0.35, progressively reducing the number of filters in each layer. The fully connected layer at the end of the original architecture has been removed. In this way, we attach the last layer of the encoder structure to a standard FCN decoder layout composed of a 2D up-sampling layer, a concatenation layer for joining the ReLU activation layers of the MobileNetV2 encoder path to two CBA blocks. The last layer adopts four 3×3 convolutional filters and a softmax activation function to generate the four probability maps, one for each labeled area of the brain tumor and the background. Final outputs of the network are obtained thanks to the argmax function applied to the $240 \times 240 \times 4$ output stack.

3) V-Net: has a similar but simplified and smaller structure compared to the U-Net. We have five stacks for the encoder and decoder paths. Each stack is made of a CBA block followed by a max pooling layer and a 1% dropout layer. In the decoder stacks, the 2D up-sampling and the concatenation layers are followed by a CBA block. The last layer has two or three 3×3 2D convolutional filters depending on whether we want to segment the whole prostate in a single class or if we want different segmentations for the central gland and the peripheral zone. As for the other segmentation networks, the last convolutional filters are followed by a softmax activation function to generate two or three probability maps. Finally, an argmax activation function is applied to the $256 \times 256 \times 2(3)$ output stack to obtain the segmented outputs.

4) 2D CNN: has a structure similar to ResNet [32] including skip connections, but it does not present the fully connected layer at the end. We have five stacks, each made of 4 CBA plus 20% dropout (CBAD) blocks, a skip layer, and a fifth CBAD block. The first stack has eight filters which double at each stack until 128. At the end of the CNN, there is a stack with 30 filters and a final 2D convolutional layer with four filters, one for each type of cell, which reduces the output to a $1 \times 1 \times 4$ vector after a softmax activation function. The predictions are obtained using the argmax activation function applied to the output vector.

Finally, all the models employed for the segmentation task were trained with a weighted focal Tversky loss [28], while the 2D CNN employed for the classification was trained with the sparse categorical cross-entropy loss. We adopted a learning rate with a step decay schedule for training both segmentation and classification networks starting from $1e-3$ and decreasing to $1e-6$, while Adam optimizes the training process [33].

IV. Experimental Results

We evaluate the DL models' performance for the different segmentation and classification tasks, comparing them to their GPU-based counterparts in terms of FPS, EE, and global DSC. Table I reports the obtained results and those of our previous work [34] (where the Vitis AI Runtime (VART) is exploited instead of PYNQ-DPU) and its projection on the new tasks. Section IV-A and Section IV-B describe the results obtained with PYNQ-DPU, while the overhead introduced by PYNQ-DPU compared to VART is discussed in Section IV-C.

A. Semantic Segmentation Evaluation

We now analyze each segmentation dataset's results ending with a general trend discussion.

1) CT-ORG: Table I shows how the Tesla V100 is the device reaching higher throughput, while both the Kria KV260 and the ZCU104 can deliver more FPS then the mobile RTX 2060. The fastest FPGA, the ZCU104, can segment a whole body scan (700 - 900 slices) in approximately 8.4 - 10.9 seconds. On the other hand, the slowest device, the Ultra96, can still provide an acceptable throughput for real-time scenarios while requiring the lowest amount of energy among all the GPUs and FPGAs. Additionally, as shown in Figure 3, all the FPGAs are more energy efficient than the GPUs; in particular, the Kria KV260 is $2.44\times$ and $6.59\times$ more energy efficient than the Tesla V100 and RTX 2060, respectively. Moreover, the quantization has brought negligible accuracy losses, with the INT8 model losing less than 0.15% DSC points, as supported by the indistinguishable segmentation differences in the first row of Figure 4.

2) BraTS: Again, Table I shows how the Tesla V100 provides the highest throughput among all the devices. For this network and task, the RTX 2060 is also faster than the FPGAs, while the Ultra96 keeps the lowest power consumption, but it has a lower energy efficiency than the Tesla V100, being the only case where an FPGA configuration is less energy efficient than a GPU one, as shown in Figure 3. The Kria KV260 is the most energy efficient, with an improvement of $1.50\times$ and $2.19\times$ upon the Tesla V100 and the RTX 2060, respectively. It also delivers higher throughput than the ZCU104, segmenting the three areas of the brain tumor in roughly 1.42 seconds per

TABLE I: Results evaluation of NERONE divided by task, dataset, and FPGA-powered board. We report the FPS, Watt, EE, and the performance metric as DSC for the semantic segmentation task and the overall accuracy (Acc) for the classification. All the results are reported as $\mu \pm \sigma$ over five independent runs, with the best in bold, underlined and italic if projected.

Task	Dataset	Device	FPS	Watt	EE	DSC/Acc
Segmentation	CT-ORG [19], [20]	RTX 2060	72.09±0.46	78.17±0.68	0.92±0.01	**90.23±0.01**
		Tesla V100	199.36±1.77	80.38±1.31	2.48±0.02	**90.23±0.01**
		Ultra96-V2	36.52±0.05	**9.59±0.03**	3.81±0.01	90.10±0.29
		Kria KV260	74.70±0.07	12.33±0.11	6.06±0.01	90.10±0.29
		ZCU104	82.92±0.47	17.59±0.02	4.71±0.01	90.10±0.29
		ZCU104†	**335.40±0.34**†	28.40±0.02†	**11.81±0.02**†	90.10±0.29
	BraTS [21]–[23]	RTX 2060	99.98±0.66	28.06±0.11	3.56±0.01	**83.44±0.38**
		Tesla V100	**372.52±3.53**	71.58±0.79	5.20±0.03	**83.44±0.38**
		Ultra96-V2	37.26±0.40	**8.96±0.11**	4.16±0.02	82.89±0.37
		Kria KV260	96.52±0.07	12.39±0.11	**7.79±0.02**	82.89±0.37
		ZCU104	93.34±0.47	16.47±0.08	5.67±0.01	82.89±0.37
		ZCU104⋆	*377.53±0.34*⋆	26.60±0.02⋆	*14.20±0.01*⋆	82.89±0.37⋆
	Prostate-3T [24]	RTX 2060	299.62±1.5	77.22±1.51	3.88±0.02	**91.81±1.31**
		Tesla V100	**599.48±3.46**	81.82±1.31	7.33±0.04	**91.81±1.31**
		Ultra96-V2	91.47±0.44	**8.59±0.03**	10.65±0.05	91.59±1.23
		Kria KV260	145.76±1.00	11.38±0.33	**12.81±0.09**	91.59±1.23
		ZCU104	190.68±0.55	15.96±0.15	11.94±0.03	91.59±1.23
		ZCU104⋆	*771.27±0.34*⋆	25.78±0.02⋆	*29.92±0.03*⋆	91.59±1.23⋆
Classification	BCCD [25]	RTX 2060	959.50±2.98	80.28±0.64	11.95±0.04	**92.25±2.25**
		Tesla V100	**2046.22±8.63**	166.56±4.29	12.29±0.05	**92.25±2.25**
		Ultra96-V2	114.99±2.18	**9.28±0.31**	12.40±0.23	92.05±2.23
		Kria KV260	237.92±0.23	12.75±0.05	**18.66±0.02**	92.05±2.23
		ZCU104	225.49±0.74	17.35±0.01	13.00±0.04	92.05±2.23
		ZCU104⋆	912.06±0.34⋆	28.02±0.02⋆	*32.55±0.04*⋆	92.05±2.23⋆

† Experimental Results from Ref. [34] ⋆ Projected according to direct VART exploit as in Ref. [34] and scaled based on † results without preprocessing

scan. Finally, the quantization procedure introduces a 0.55% drop in DSC, which, despite being the more significant drop in accuracy among all the datasets, is negligible to the human eye, as in the central row of Figure 4.

3) Prostate-3T: Table I shows how the behavior of the FP32 and INT8 networks remains similar to the previous two tasks but with a generally higher EE (as in Figure 3). The higher EE is the effect of a higher throughput without a proportional power consumption growth. The Tesla V100 remains the most performing in FPS, requiring roughly the same power as the RTX 2060 but providing 2× higher throughput. The ZCU104 achieves 190 FPS, being the fastest among the FPGAs and allowing for segmenting the prostate area in an average of 0.1 seconds. Though the Ultra96 has the lowest power consumption, its EE is lower than the others keeping the Kria KV260 as the most energy-efficient device. It has an improvement of 1.74× and 3.3× compared to the Tesla V100 and RTX 2060 configurations. The quantized version of the network has just a 0.21% drop in the DSC, which is also visually confirmed by the visual outputs in the bottom row of Figure 4.

Finally, to summarize the main trends for semantic segmentation, as expected, the Tesla V100 is the fastest being a datacenter card. Regarding power consumption, FPGAs are always better than GPUs, as shown in Table I. Conversely, in one case, namely on BraTS in Figure 3, the Ultra96 is less energy efficient than the V100, although this is imputable to the much higher FPS rate of the latter. Ultimately, the Kria KV260 proved the best trade-off between watts and FPS.

B. Multi-Class Classification Evaluation

Moving to the classification task, we consider the same metrics besides the DSC, which is substituted by the accuracy.

1) BCCD: Given the network structure and the simpler task compared to the semantic segmentation, as in Table I, the Tesla V100 reaches more than 2000 FPS (with more than 166.56 Watt), while the RTX 2060 reaches almost 1000 FPS. Also, in this case, the Ultra96 is less power-hungry with significant energy efficiency, while the Kria KV260 outperforms all the other four configurations, as shown in Figure 3, with a $1.52\times$ and a $1.56\times$ improvements in energy efficiency. These limited improvements are related to the massive difference in the FPS ($\sim 4.1 - 8.6\times$ more) the GPUs can reach compared to the FPGAs. On the other hand, if we consider a constrained scenario, none of the GPU solutions is considerable in terms of required watts per computation. Finally, the quantized version of the CNN has negligible accuracy losses with just a 0.08% drop compared to the GPU counterpart.

C. PYNQ-DPU Overhead

As Section II-C described, PYNQ-DPU eases the development process for the end users, but as shown in Table I, at the

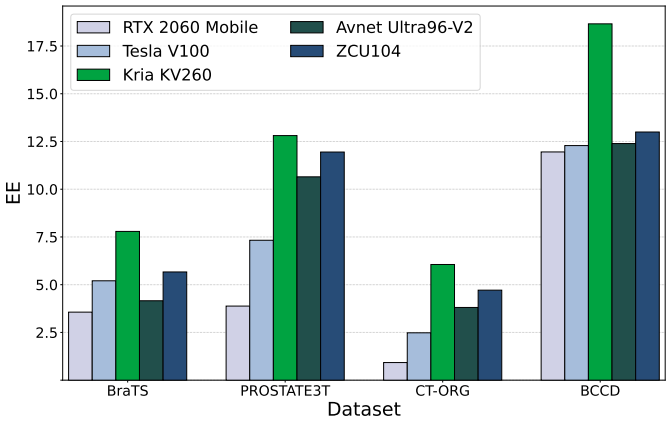

Fig. 3: Average Energy Efficiency (EE) of each tested device clustered by dataset.

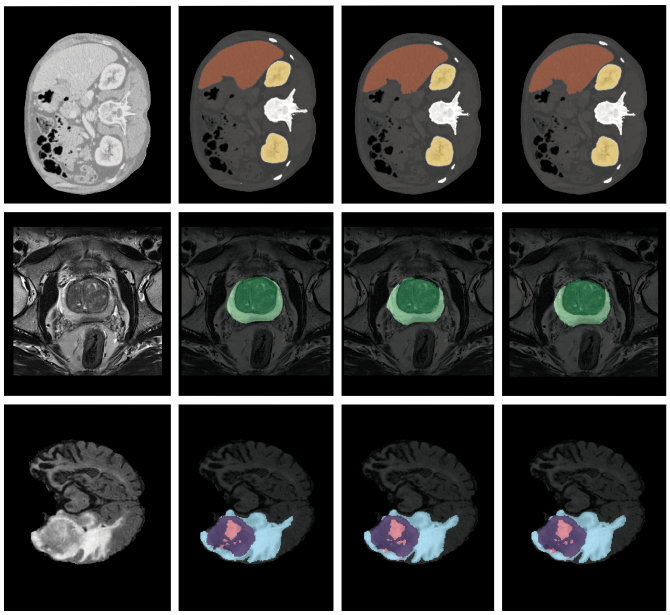

Fig. 4: Visual comparison of NERONE results. From left to right: input image, ground truth, INT8 NERONE segmentation, FP32 NERONE segmentation. Top to bottom: the CT-ORG [19], [20] dataset with spinal cord in white, kidneys in yellow and liver in brown. From the Prostate-3T [24] dataset, the inner central gland in dark green and the peripheral zone in light green. From the BraTS [21]–[23] dataset, the outer peritumoral edema in light blue, the enhancing tumor in purple, and the pink tumor core.

cost of impacting FPGA performance. To evaluate such an impact, we tested the ZCU104-based solution on the CT-ORG dataset of our previous work [34] employing the default Linux-Vitis AI Runtime (VART) image and reported the projections of the results on the other datasets. Table I shows how using VART outperforms all the PYNQ-DPU configurations and the GPU ones in terms of FPS and EE. Notably, PYNQ-DPU data accounts for data preprocessing time, while with VART, the data should be preprocessed offline.

Looking at the CT-ORG dataset VART on the ZCU104 is $4.04\times$ faster than our PYNQ-DPU one. Moreover, it is also $1.68\times$ faster than the high-end class Tesla V100 which consistently outperformed the PYNQ-DPU FPGAs throughput. Despite the increase in watt consumption introduced by VART, the configuration still presents a higher energy efficiency of $1.95\times$ better than the one of the Kria KV260, which is the most energy-efficient device tested within NERONE. Even though VART provides a performance improvement, it leads to a less versatile and agile use of the FPGAs. A simple example is the absence of a package manager that leads, for instance, to be unable to install extra Python packages. On the other hand, PYNQ-DPU provides more tools to directly develop and test the solution on the FPGAs, such as a package manager and a Jupyter Notebook that can be used via a graphical user interface. To sum up, NERONE, thanks to PYNQ-DPU, can efficiently quantize, compile and deploy a working application on multiple FPGAs with little effort by the user. On the other hand, VART assures performance maximization and is the way to go when the edge execution environment is stable.

V. Literature Analysis and Comparison

This Section compares NERONE against literature works that exploit the same datasets in our experimental evaluation. Table II reports our best performing FPGA and GPU configurations, namely the Kria KV260 and the Tesla V100. On the CT-ORG dataset, [35] combines two datasets to increase the data available to train a 3D U-Net and a 3D DenseV-Net. They showed how segmentation performance would increase on organs labeled on multiple datasets while losing some accuracy on those that appear only in one of the two. The 3D U-Net is also employed by [20] coupled with a smoothed IOU loss. For training and inference, they employ 4 GPUs (whose model is not specified), achieving an average of 4.3 s to segment a CT scan (17-197 FPS). On the BraTS dataset, [36] conducts some experiments using a 3D U-Net by testing combinations of region-based training, data augmentation, and batch loss instead of sample loss, while [37] employs an encoder-decoder architecture but with scale attention blocks connecting each encoding layer output to all the decoding layers obtaining better DSCs compared to plain U-Net. For the Prostate-3T dataset, [39] employs a DeepLabV3+ to show the improvement obtained by the atrous convolution compared to classical FCN, while [38] is the subsequent work where the authors improve the DeepLabV3+ performance with patch-wise training. On the BCCD dataset, [40] employed a pre-trained MobileNet-224 as a feature extractor followed by a logistic regressor as the final classifier, while [41] employs a small convolutional neural network with a fully connected layer that includes more than 97% of the network total parameters, achieving better results than fine-tuning on pre-trained networks.

An interesting point to note is that the literature does not consider throughput or energy efficiency while evaluating performance. Indeed, all the selected works' primary objective is to reach the most accurate network on a specific dataset and task without focusing on the final application, that in the medical field, along with accuracy, requires further consideration on the constraints of the working environment.

TABLE II: Comparison of NERONE with other works in the literature dealing with the same dataset employed in the NERONE evaluation. We report FPS, EE, and specific task related performance metrics. All our results are reported as $\mu \pm \sigma$ over five independent runs, while we report the numbers in the paper of the literature works. The best results are in bold.

Task	Dataset	Metric	Our Kria KV260	Our Tesla V100	[35]	[20]
Segmentation	CT-ORG [19], [20]	FPS	74.70±0.07	**199.36±1.77**	>6.4	[17-197]
		EE	**6.06±0.01**	2.48±0.02	n.a.	n.a.
		DSC Liver	91.08±0.35	90.8±0.02	**94.90±n.a.**	92.00±3.6
		DSC Bladder	77.74±0.40	74.95±0.02	**85.80±n.a.**	58.10±22.3
		DSC Lungs	92.83±0.38	93.21±0.02	**98.35±n.a.**	93.80±5.9
		DSC Kidneys	80.69±0.34	79.61±0.02	**91.25±n.a.**	88.20±7.9
		DSC Bones	89.84±0.10	90.03±0.02	**92.73±n.a.**	82.70±7.6
			Our Kria KV260	Our Tesla V100	[36]	[37]
	BraTS [21]–[23]	FPS	96.52±0.07	**372.52±3.53**	n.a.	n.a.
		EE	**7.79±0.02**	5.20±0.03	n.a.	n.a.
		DSC Whole Tumor	**93.04±0.31**	**93.09±0.53**	91.87±n.a.	92.56±1.05
		DSC Tumor Core	81.05±0.47	82.26±0.48	87.97±n.a.	**89.02±1.64**
		DSC Enhancing Tumor	**87.28±0.46**	**87.46±0.46**	81.37±n.a.	82.20±1.74
			Our Kria KV260	Our Tesla V100	[38]	[39]
	Prostate-3T [24]	FPS	145.76±1.00	**599.48±3.46**	n.a.	n.a.
		EE	**12.81±0.09**	7.33±0.04	n.a.	n.a.
		DSC Central Gland	89.05±1.39	89.05±1.47	**92.80±0.70**	88.40±n.a.
		DSC Peripheral Zone	72.01±1.33	72.82±1.39	**78.90±1.90**	70.20±n.a.
			Our Kria KV260	Our Tesla V100	[40]	[41]
Classification	BCCD [25]	FPS	237.92±0.23	**2046.22±8.63**	n.a.	n.a.
		EE	**18.66±0.02**	12.29±0.05	n.a.	n.a.
		Acc	92.05±2.23	92.25±2.25	**97.03**	87.00
		Eosinophil	**96.40±1.47**	**96.54±1.54**	96.00	83.15
		Monocyte	97.15±0.55	97.47±0.68	**100.00**	**100.00**
		Lymphocyte	97.57±1.00	97.91±0.87	**100.00**	75.00
		Neutrophil	92.97±2.34	93.20±2.51	**96.00**	93.59

Conversely, with NERONE, we provide the end users with an easy-to-use and transparent framework to deploy DL models in constrained scenarios, where energy efficiency and high throughput rates are essential. Moreover, since NERONE starts from a trained model, the users can still focus on pushing the accuracy performance to the top before transparently moving their models on FPGA. From Table II, we can also see how all the models proposed to evaluate NERONE on each task and dataset are still reaching significant results in terms of specific performance metrics even if reaching top DSC/accuracy results is not the core objective of the proposed work.

VI. Discussion and Conclusions

This Section discusses the benefits of employing NERONE to exploit an FPGA-based accelerated inference and the ease of developing custom solutions starting from the proposed framework. Indeed, we have proposed an open-source framework that, given a trained network and a dataset, transparently deploys the inference phase on FPGA, maintaining the resulting accuracy but improving the energy efficiency compared to GPU-based solutions. Moreover, thanks to NERONE, end users can benefit from the FPGA-based inference without any coding overhead, but just by writing a few lines of Python code. Based on these considerations, we believe the proposed framework could pave the way for broader adoption of FPGAs in the DL world, opening to new applications (e.g., where power consumption is a limiting factor) and a vaster public, for example, to computer vision experts.

Even though we have demonstrated the ability of NERONE to be easily adaptable to different image modalities, tasks, and platforms, further considerations should be made. While the FPGA-based embedded solutions proposed outperform the GPU ones in terms of energy efficiency, they still achieve lower FPS rates. This fact is mainly caused by the overhead of using the Python APIs provided by PYNQ-DPU. Although employing them facilitates the usage by the end users, it also introduces a computational overhead that can be avoided by directly exploiting the Vitis AI Runtime (VART).

In conclusion, within this work, we have proposed an open-source framework for deploying DL models on FPGAs to achieve a fast, accurate, and efficient inference. By its evaluation, we have demonstrated how it can be easily tailored to different use cases without knowing FPGA programming.

Acknowledgments

We thank Oracle Cloud Infrastructure and Oracle for Research for their support to this work. This work was supported in part by Oracle Cloud credits and related resources provided by the Oracle for Research program. We thank AMD University program for the boards used in this manuscript.

REFERENCES

[1] J. Hsieh, "Computed tomography: principles, design, artifacts, and recent advances," 2003.

[2] R. W. Brown, Y.-C. N. Cheng, E. M. Haacke, M. R. Thompson, and R. Venkatesan, *Magnetic resonance imaging: physical principles and sequence design*. John Wiley & Sons, 2014.

[3] M. I. Razzak, S. Naz, and A. Zaib, *Deep Learning for Medical Image Processing: Overview, Challenges and the Future*. Cham: Springer International Publishing, 2018, pp. 323–350. [Online]. Available: https://doi.org/10.1007/978-3-319-65981-7_12

[4] Y. Guo, Y. Liu, T. Georgiou, and M. S. Lew, "A review of semantic segmentation using deep neural networks," *International journal of multimedia information retrieval*, vol. 7, no. 2, pp. 87–93, 2018.

[5] E. D'Arnese, G. W. Di Donato, E. Del Sozzo, M. Sollini, D. Sciuto, and M. D. Santambrogio, "On the automation of radiomics-based identification and characterization of nsclc," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 6, pp. 2670–2679, 2022.

[6] Z. Ma, J. M. R. Tavares, and R. N. Jorge, "A review on the current segmentation algorithms for medical images," in *Proceedings of the 1st International Conference on Imaging Theory and Applications (IMAGAPP)*, 2009.

[7] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A review on deep learning techniques applied to semantic segmentation," *arXiv preprint arXiv:1704.06857*, 2017.

[8] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[9] E. Alcaín, P. R. Fernández, R. Nieto, A. S. Montemayor, J. Vilas, A. Galiana-Bordera *et al.*, "Hardware architectures for real-time medical imaging," *Electronics*, vol. 10, no. 24, p. 3118, 2021.

[10] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues *et al.*, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[11] L. Zhang, X. Wang, D. Yang, T. Sanford, S. Harmon, B. Turkbey *et al.*, "Generalizing deep learning for medical image segmentation to unseen domains via deep stacked transformation," *IEEE transactions on medical imaging*, vol. 39, no. 7, pp. 2531–2540, 2020.

[12] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, 2017, pp. 1–12.

[13] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "[dl] a survey of fpga-based neural network inference accelerators," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 12, no. 1, pp. 1–26, 2019.

[14] E. D'Arnese, D. Conficconi, M. D. Santambrogio, and D. Sciuto, "Reconfigurable architectures: The shift from general systems to domain specific solutions," in *Emerging Computing: From Devices to Systems*. Springer, 2023, pp. 435–456.

[15] S. I. Venieris, A. Kouris, and C.-S. Bouganis, "Toolflows for mapping convolutional neural networks on fpgas: A survey and future directions," *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, pp. 1–39, 2018.

[16] Xilinx, "Vitis ai: Adaptable and real-time ai inference acceleration," https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html, 2021, last accessed 21 January 2022.

[17] ——, "Dpu for convolutional neural network," https://www.xilinx.com/products/intellectual-property/dpu.html#overview, 2021, last accessed 30 April 2023.

[18] X. Zhang, J. Wang, C. Zhu, Y. Lin, J. Xiong, W.-m. Hwu *et al.*, "Dnnbuilder: an automated tool for building high-performance dnn hardware accelerators for fpgas," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2018, pp. 1–8.

[19] Blaine Rister, Kaushik Shivakumar, Tomomi Nobashi and Daniel L. Rubin, "Ct-org: Ct volumes with multiple organ segmentations [dataset]," *The Cancer Imaging Archive*, 2019. [Online]. Available: https://doi.org/10.7937/tcia.2019.tt7f4v7o

[20] B. Rister, D. Yi, K. Shivakumar, T. Nobashi, and D. L. Rubin, "Ct-org, a new dataset for multiple organ segmentation in computed tomography," *Scientific Data*, vol. 7, no. 1, pp. 1–9, 2020.

[21] U. Baid, S. Ghodasara, S. Mohan, M. Bilello, E. Calabrese, E. Colak *et al.*, "The rsna-asnr-miccai brats 2021 benchmark on brain tumor segmentation and radiogenomic classification," *arXiv preprint arXiv:2107.02314*, 2021.

[22] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby *et al.*, "The multimodal brain tumor image segmentation benchmark (brats)," *IEEE transactions on medical imaging*, vol. 34, no. 10, pp. 1993–2024, 2014.

[23] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. S. Kirby *et al.*, "Advancing the cancer genome atlas glioma mri collections with expert segmentation labels and radiomic features," *Scientific data*, vol. 4, no. 1, pp. 1–13, 2017.

[24] Litjens, Geert, Futterer, Jurgen, Huisman, and Henkjan, "Data from prostate-3t [data set]," *The Cancer Imaging Archive. Available online: 0.7937/K9/TCIA.2015.QJTV5IL5*, 2015.

[25] S. Cheng and N. Chen, "Blood Cell Count and Detection," 2019. [Online]. Available: https://github.com/Shenggan/BCCD_Dataset

[26] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[27] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 fourth international conference on 3D vision (3DV)*. IEEE, 2016, pp. 565–571.

[28] R. Berzoini, A. A. Colombo, S. Bardini, A. Conelli, E. D'Arnese, and M. D. Santambrogio, "An optimized u-net for unbalanced multi-organ segmentation," in *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, 2022, pp. 3764–3767.

[29] 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.

[30] W. Sae-Lim, W. Wettayaprasit, and P. Aiyarak, "Convolutional neural networks using mobilenet for skin lesion classification," in *2019 16th international joint conference on computer science and software engineering (JCSSE)*. IEEE, 2019, pp. 242–247.

[31] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, 2021.

[32] 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.

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

[34] R. Berzoini, E. D'Arnese, and D. Conficconi, "On how to push efficient medical semantic segmentation to the edge: the seneca approach," in *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2022, pp. 1–8.

[35] F. Islam Tushar, H. Nujaim, W. Fu, E. Abadi, M. A. Mazurowski, E. Samei *et al.*, "Quality or quantity: Toward a unified approach for multi-organ segmentation in body ct," *arXiv e-prints*, pp. arXiv–2203, 2022.

[36] F. Isensee, P. F. Jäger, P. M. Full, P. Vollmuth, and K. H. Maier-Hein, "nnu-net for brain tumor segmentation," in *International MICCAI Brainlesion Workshop*. Springer, 2020, pp. 118–132.

[37] Y. Yuan, "Automatic brain tumor segmentation with scale attention network," in *International MICCAI Brainlesion Workshop*. Springer, 2020, pp. 285–294.

[38] Z. Khan, N. Yahya, K. Alsaih, S. S. A. Ali, and F. Meriaudeau, "Evaluation of deep neural networks for semantic segmentation of prostate in t2w mri," *Sensors*, vol. 20, no. 11, p. 3183, 2020.

[39] Z. Khan, N. Yahya, K. Alsaih, and F. Meriaudeau, "Zonal segmentation of prostate t2w-mri using atrous convolutional neural network," in *2019 IEEE Student Conference on Research and Development (SCOReD)*. IEEE, 2019, pp. 95–99.

[40] E. H Mohamed, W. H El-Behaidy, G. Khoriba, and J. Li, "Improved white blood cells classification based on pre-trained deep learning models," *Journal of Communications Software and Systems*, vol. 16, no. 1, pp. 37–45, 2020.

[41] M. Sharma, A. Bhave, and R. R. Janghel, "White blood cell classification using convolutional neural network," in *Soft Computing and Signal Processing*. Springer, 2019, pp. 135–143.