On How to Push Efficient Medical Semantic Segmentation to the Edge: the SENECA approach

Raffaele Berzoini, Eleonora D'Arnese, Davide Conficconi
Politecnico di Milano, Piazza Leonardo da Vinci 32, Milano, Italy
raffaele.berzoini@mail.polimi.it, {eleonora.darnese, davide.conficconi}@polimi.it

***Abstract*—Semantic segmentation is the process of assigning each input image pixel a value representing a class, and it enables the clustering of pixels into object instances. It is a highly employed computer vision task in various fields such as autonomous driving and medical image analysis. In particular, in medical practice, semantic segmentation identifies different regions of interest within an image, like different organs or anomalies such as tumors. Fully Convolutional Networks (FCNs) have been employed to solve semantic segmentation in different fields and found their way in the medical one. In this context, the low contrast among semantically different areas, the constraint related to energy consumption, and computation resource availability increase the complexity and limit their adoption in daily practice. Based on these considerations, we propose SENECA to bring medical semantic segmentation to the edge with high energy efficiency and low segmentation time while preserving the accuracy. We reached a throughput of 335.4 ± 0.34 frames per second on the FPGA, $4.65\times$ better than its GPU counterpart, with a global dice score of $93.04\% \pm 0.07$ and an improvement in terms of energy efficiency with respect to the GPU of $12.7\times$.**

***Index Terms*—Semantic Segmentation, Hardware Acceleration, U-Net, FPGA**

I. Introduction

Semantic segmentation is a widespread technique in computer vision that segments different objects inside a given image, relying on a pixel-level classification [1]. It finds application in a vast range of fields from autonomous driving and robotic navigation to medical practice, where the understanding of the observed scene passes through the clustering of the different pixels into homogeneous areas representing different objects or semantic classes [2], [3]. This task is nowadays resolved thanks to the adoption of Deep Learning (DL) [4] and, more precisely, thanks to Fully Convolutional Networks (FCNs) [5]. FCNs output for each image input pixels a value indicating its class, and, therefore, provide a fast and easy way to interpret information map regarding the observed scene. Thanks to their vast development in other fields, FCNs have found a fertile field in medical practice [6]–[8], where the low contrast among different object classes complicates the semantic segmentation task. Indeed, medical imaging relies on gray-scale images that present different areas or organs with similar intensities and with low contrast in the border regions. For these reasons, through the years, medical semantic segmentation was performed primarily manually or in a semi-automatic fashion by experts, requiring a lengthy analysis time [9]. Since the explosion of interest related to artificial intelligence, clinical practice has also introduced new automatic computer-based solutions. In this direction, simplifying the semantic segmentation task through DL-based solutions [4] would positively impact the analysis time of numerous medical imaging tasks.

Unfortunately, the deployment of such solutions requires computational and energetic resources that are not always available in the medical field. Indeed, the deployment of semantic segmentation DL models requires a vast and annotated dataset to be trained on and a powerful computational system to run on. With these constraints, given that a hospital generally has a considerable amount of pre-existing data already analyzed, the main limiting factor remains the computational configuration. The most employed devices for running FCNs are GPUs which deliver notable performances in throughput of segmented images but at the cost of high energy consumption. Such a need limits this configuration in a surgery scenario where we want to perform the semantic segmentation of images acquired in real-time on the surgery table where the surgery and imaging pieces of machinery require energy. For these reasons, there are alternative solutions employing either domain-specific architectures (DSAs) [10] or FPGAs for their reconfigurable fabrics [11]. While the firsts are frozen in silicon, FPGA-based ones provide continuously updatable solutions dividable into two categories. The first one embeds the FCN structure directly in the fabric through automation toolchains [12] while the other implements soft-DSAs [13], [14] that are software configurable for a wide range of FCNs. These latest engines represent the most flexible solution and exploit domain specialization being software-programmable, reconfigurable at hardware level, and tailored for convolutions.

Within this context, we propose SENECA, a quantized semantic segmentation network, running on FPGA, able to accurately segment five major abdominal organs, namely lungs, liver, bones, kidneys, and bladder, from the publicly available dataset CT-ORG [15]–[17]. We employ Vitis AI comprehensive stack [18] to develop a systematic workflow for deploying energy-efficient semantic segmentation network on Xilinx DPU [14] on an edge device, namely the ZCU104. SENECA reaches a $4.65\times$ speedup and a $12.7\times$ reduction in

DOI 10.1109/IPDPSW55747.2022.00027

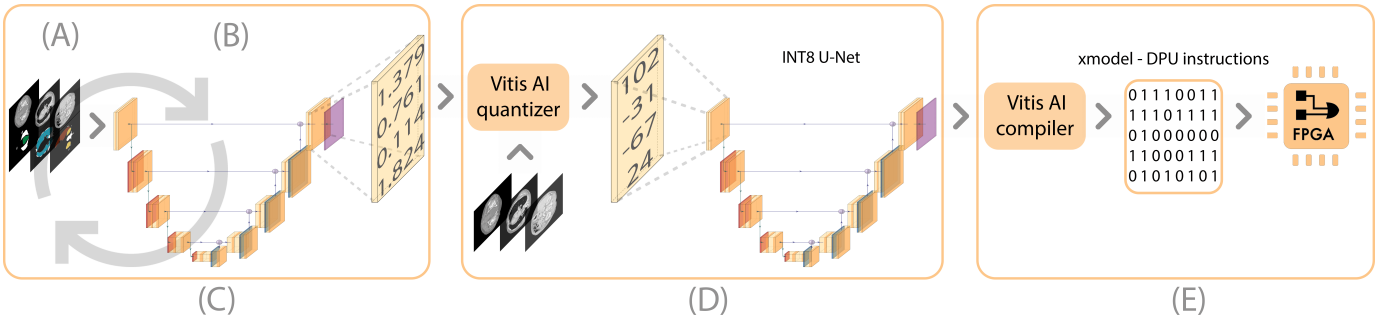

Fig. 1: High-level scheme of the overall proposed framework, which consists of data preparation and pre-processing (A), 32-bit floating-point weights U-Net definition (B), and its subsequent training (C). Then, it proceeds with model quantization via Vitis AI quantizer and calibration dataset to obtain the 8-bit integer U-Net version (D). Finally, the Vitis AI compiler generates DPU-optimized instructions to deploy the quantized model on the FPGA (E).

energy consumption compared to its GPU-based counterpart. Finally, it improves the accuracy compared to the CT-ORG work [17], reducing its variability. Our main contributions are:

- The novel tailoring of real medical image semantic segmentation models to a reconfigurable edge device.
- The development of an open-source[1] methodological workflow for deploying efficient medical semantic segmentation models, easily adaptable to other DL networks.
- A systematic design and results analysis to identify the best semantic segmentation model.

The remainder of the paper is organized as follows: Section II presents literature works in the semantic segmentation and acceleration. Section III details the proposed solution implementation choices, while Section IV describes the experimental setup and the SENECA evaluation in terms of accuracy, inference time, and energy efficiency. Finally, Section V draws the conclusions and discusses possible future improvements.

II. Related Work

Machine Learning (ML), specifically DL and FCNs, have won the so-called *"hardware lottery"* [19] of being a good research idea that wins for compatibility with available hardware and software. Therefore a considerable body of research has been done on efficient ML inference on specific hardware.

GPUs are the de-facto standard for DL training. However, their power-hungry nature opens to different inference engines. Although silicon development is a considerable cost that is not always viable, many big companies are developing their custom DSA for efficient FCN execution, such as the Google Tensor Processing Unit [10]. Besides, FPGAs are the alternative to GPUs and DSAs. Their reconfigurable fabric and fine-grained configurability made them the candidate platform for energy-efficient quantized DL models execution [11]. In a first instance, many FPGAs experts developed custom architectures for specific DL networks embedded in the FPGA logic [20], [21]. To lower the learning curve barrier, researchers develop design automation toolchains for Convolutional Neural Networks (CNNs) [12]. Hybrid solutions between silicon-based DSA and logic embedding are soft-DSAs [13], [14] which adopt the software-programmability of the former and the hardware reconfigurability of the latter while keeping domain-specialization. We adopt this latter engine category for our semantic segmentation workflow, being the optimal trade-off between flexibility and specialization.

[1]https://github.com/RaffaeleBerzoini/SENECA.git

TABLE I: Organ frequencies in the CT-ORG dataset, expressed as pixel percentage of labeled targets.

Liver	Bladder	Lungs	Kidneys	Bones	Brain
22.18%	2.51%	34.17%	4.70%	36.26%	0.18%

On the semantic segmentation side, there are various approaches to be considered. The first one is proposed along with the CT-ORG dataset [15]–[17], which employs a 3D U-Net to segment the different labeled organs. Exploiting other datasets, reference [22] employs a DenseVnet reaching high accuracy for small organs but less impressing results on the bigger ones. Differently, the Organ Attention Network (OAN) in [23] segments the labeled organs along the three principal axes and then combines the obtained maps with a statistical function, reaching remarkable performance. Unfortunately, its adoption is limited by the necessity of having organs adjacency.

Based on all these considerations, we exploit Vitis AI [18] open-ecosystem and its soft-DSA called Deep Learning Processor Unit (DPU) [14] to deliver high-performance energy-efficient semantic segmentation inference with a 2D U-Net-based model. Moreover, to the best of the authors' knowledge, this is the first solution dealing with medical semantic segmentation at the edge exploiting an FPGA in a soft-DSA fashion.

III. Proposed Approach

This Section describes the systematic workflow for inference acceleration of the SENECA semantic segmentation FCN based on Vitis AI [18]. Figure 1 shows the main proposed steps from the employed dataset, the considered 2D model and its training to its optimization, and the deployment of the SENECA model on the edge device, namely the ZCU104.

TABLE II: Number of layers, filters and total parameters of each implemented and compared model in this work.

Configuration	Layers	Filters	Parameters $[\times 10^6]$
1M	9	8	~ 1.034
2M	11	6	~ 2.329
4M	11	8	~ 4.136
8M	11	11	~ 7.814
16M	11	16	~ 16.522

A. Data Description and Pre-processing

We start the description of SENECA with some considerations on the data employed that have affected some of the implementation decisions. We exploit the open source CT-ORG dataset [15]–[17] to develop our semantic segmentation workflow. It comprises 140 CT total-body or chest-only acquisition, one per patient, saved in NIfTI format, with a variable bit-width ranging from 16 to 32. For each volume, CT-ORG provides a corresponding volume reporting the ground truth label for six organs, namely bladder, bones, brain, kidneys, liver, and lungs. Table I reports the frequency of each organ in the dataset, and we can notice how it correctly reflects the biological size of the considered organs. The only exception is the brain, which proves to be highly underrepresented; indeed, it has a similar dimension to the liver but appears only to be 0.18% of the total labeled pixels in the dataset. Such a disproportion derives from the low number of total-body CTs and makes us remove the brain from the target organs. In addition, we downsized the input images from 512×512 to 256×256 and rescaled them in the [-1, 1] interval to better suit the further implementation steps. As the final pre-processing step, we adjust the contrast of the images by saturating the upper 1% and the lower 1% of the pixels.

B. 2D U-Net Model

The model chosen as the skeleton for SENECA is a 2D FCN based on a U-Net architecture [24] that allows us to preserve the spatial information in the output. We prefer a 2D structure to a 3D one since it is faster to train and requires less memory without losing accuracy. We have implemented five models varying the number of layers and filters to identify the best-suited model to semantically segment the CT data based on throughput, energy consumption, and accuracy. Table II shows the considered configurations, highlighting the different structures and the overall number of trainable parameters. The high-level structure of the proposed models reflects the U-Net one with an encoding-decoding shape, as in Figure 1. Each encoder stack consists of two 3×3 convolutional layers, doubling the number of filters going downward, a batch-normalization followed by a rectified linear unit (ReLU) activation function, and ends with a 2×2 max pooling operation for down-sampling and a dropout layer to prevent overfitting. On the other hand, each decoder stack is similar to its encoder counterpart with the addition of a 3×3 transpose convolution for up-sampling and a concatenation operation to merge the feature maps coming from the encoders into the decoder ones. Each decoder stack halves the number of filters in the convolutional operations. The last layer employs six 3×3 convolutional filters and a softmax activation operation to produce six probability maps, one for each of the five organs and the sixth for the background. Finally, the output predicted labels are obtained using an argmax function on the $256 \times 256 \times 6$ output stacks.

C. Training Loss Function

As stated in Section III-A, although we removed the brain labels, the dataset still presents a class imbalance problem deriving from the human anatomy, where lungs and liver are the largest organs along with bones that appear in almost each image, while kidneys and bladder are the smallest ones. This dimensions variability leads the model to focus more on easy target compared to the less frequent examples during training. To overcome the class imbalance problem we implement a weighted loss function to force the training towards the most difficult examples. In particular, we exploit the Focal Tversky loss, which proves to be the best candidate for the considered model and dataset. We define the loss as:

$$FTL_w = (1 - (\frac{\sum_c w_c \cdot TI_c}{\sum_c w_c}))^{\gamma} \tag{1}$$

where TI_c and w_c are the Tversky Index and the weight assigned to the class c, respectively, and γ is the parameter that regulates the training towards easier or harder examples.

In particular, TI_c is computed as:

$$TI_c = \frac{\sum_{i=1}^{N} p_{ic} g_{ic}}{\sum_{i=1}^{N} p_{ic} g_{ic} + \alpha \sum_{i=1}^{N} \bar{p}_{ic} g_{ic} + \beta \sum_{i=1}^{N} p_{ic} \bar{g}_{ic}} \tag{2}$$

where p_{ic} and $\bar{p}_{ic}$ are respectively the predicted probability that pixel i belongs to class c or not; g_{ic} is the ground truth and it is equal to 1 for pixel belonging to class c while $\bar{g}_{ic}$ is equal to 1 for pixel not belonging to class c; α and β are the regularization parameters for false negative and false positive, respectively. We set $\alpha = 0.7$ and, $\beta = 0.3$ which is the best combination according to [25].

Finally, when in Equation (1) $\gamma > 1$, the Focal Tversky loss pushes the network's training towards the most difficult examples (i.e., the organs associated with a lower Tversky Index). Therefore, we set $\gamma = \frac{4}{3}$, being in the suggested range $[1 - 3]$ [26], to produce a first mitigation of the dataset class imbalance problem. Secondly, we assign to each TI_c a custom weight w_c inversely proportional to the organ dimensions. Indeed, the bigger organs are associated with small weight values, while smaller organs present higher weights.

D. Model Quantization

Generally, training is performed with at least 32-bit floating-point (FP32) weights, as we have done in this work. However, while many preserve FP32 at inference time, quantization is applied to reduce the memory footprint and save energy from the computation. As many state-of-the-art works demonstrated [11], [27], [28], employing weights with 8-bit integer (INT8)

TABLE III: Organ frequencies in the calibration data set before (Random Sampling) and after (Manual Sampling) manual organs frequencies correction.

	Liver	Bladder	Lungs	Kidneys	Bones
Random Sampling	24.38%	3.00%	35.27%	3.63%	33.72%
Manual Sampling	21.69%	7.66%	32.02%	6.90%	31.73%

precision drastically reduce the required memory and energy with an (often) negligible impact on the accuracy. Indeed, the Vitis AI quantizer can convert FP32 weights to INT8 without degrading the resulting accuracy. In this way, the final execution engine, i.e., the DPU, can execute a faster INT8-weights model in an energy-efficient way, thanks also to a reduced memory bandwidth demand. On top of this, the quantization tool folds batch-normalization layers and removes nodes not required for inference, such as dropout layers, further reducing the overall computation and memory demand. Vitis AI quantizer tool provides three quantization procedures: *Post Training Quantization (PTQ)*, *Fast Finetuning Quantization (FFQ)*, and *Quantization Aware Training (QAT)*.

PTQ applies quantization through a small (100-1000 images) unlabeled calibration dataset. Whenever PTQ impacts the model with accuracy loss, FFQ can reduce such loss by increasing the quantization time required. Indeed, FFQ is based on the AdaQuant algorithm [29] that adjusts weights and quantizes parameters layer-by-layer using a calibration dataset composed of unlabeled images. Finally, QAT can apply a drastic quantization technique. Indeed, through this method, the Vitis AI quantizer rewrites the floating graph and converts it to a quantized model before network training. However, this method requires the whole training dataset (input and labeled images) and longer computational times compared both to PTQ and standard GPU model training.

We opt for the first method, i.e., PTQ. For all the chosen models listed in Table II, this quantization method generates a quantized model with no global performance losses with a calibration dataset of 500 images. We decide to test both the remaining FFQ and QAT, but without achieving improvements over PTQ. Afterward, the quantizer tool exploits the calibration dataset to identify modifications applied in the FP32 to INT8 weights transformation. With the information gathered, the tool reduces inference differences between FP32 and INT8 models as much as possible. Since these adjustments are based on a portion of the original training dataset, the distribution of the pixels belonging to the organs remains similar to Table I.

Because of this imbalanced distribution, weights transformation tries to minimize the changes during inference in the most frequent cases, and it lets sporadic cases (such as the bladder) contribute very little to weights transformation and adjustment. An intuitive and quite effective way to counter the loss of performance on smaller organs is to build the calibration dataset to level out organs frequency. However, we found that an excessive change in the original pixel distribution led to relevant global performance losses. We experimented with several combinations of pixels distributions and found

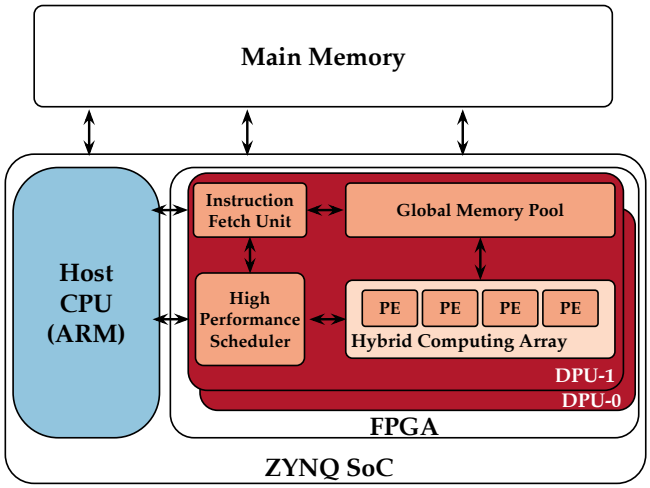

Fig. 2: System view of the Zynq-based dual core DPUCZDX8G-B4096 highlighting its internal architecture and its system-wise components.

the reported distribution in Table III to be the best. This distribution allows a model quantization with tiny performance loss on some organs but better generalization, with slight performances improvement, on some others. All things considered, we obtained a small global accuracy improvement supported by a remarkable throughput increase.

E. Model Compilation and Deployment to the Edge

After model training and quantization steps, we are ready for network compilation, deployment, and evaluation on a target FPGA, a Xilinx ZCU104 evaluation board in our case. Before such procedures, we need to compile the model into binary instructions, namely an *xmodel* file, of our target programmable soft-engine, which is the DPU. Figure 2 reports a system view of the considered edge device composed of a system-on-a-chip (SoC) with the main memory off-chip. The SoC is composed of an ARM-based CPU and an FPGA, on which we implement our target DPU. The default ZCU104 configuration is a dual-core DPUCZDX8G-B4096, which is, among the DPU configurations, a general-purpose DPU for CNNs that employs INT8 quantized data containers. It employs three parallelism degrees called pixel, input channel, and output channel parallelism, whose multiplication determines the peak operations per clock cycle of 4096.

We took advantage of the Vitis AI compiler, called VAI_C, to perform the final xmodel translation. The VAI_C framework parses the topology of the quantized input model and constructs an internal computation graph. As in the quantization step, VAI_C performs multiple compile-time optimizations. For instance, batch normalization nodes are fused into preceding convolutional layers, and efficient instruction scheduling by exploiting parallelism and data reuse is produced. Then, the back-end generates the compiled xmodel file based on the target DPU microarchitecture.

Now, we are ready to deploy our compiled xmodel to the target FPGA. We then employ the Vitis AI Runtime (VART) to asynchronously submit and collect jobs to/from the accelerator and take advantage of our multi-core architecture. Generally, VART enables C++ and Python APIs (here exploited) to preprocess the input images and start the multithreading inference process on the ZCU104. In contrast to the original FP32-weights model trained on FP32 images to generate FP32 segmentations, we now have to account that our compiled model is INT8. Therefore, it only accepts INT8 images as input and returns INT8 masks of the segmented organs. Within this context, we scaled input slices with a specific factor generated during compilation and stored into the xmodel.

IV. Experimental Results

This Section discusses the performance of SENECA. We show and analyze the results from different points of view to better understand and compare all the configurations tested for each model. Then, we select the best configuration among all the ones presented, and we will further analyze it by exploiting more metrics and comparing it with the CT-ORG work [17].

A. Experimental Setup

We develop SENECA by constructing and training the five proposed models. We exploit TensorFlow 2 for the network structures definition, and we train them on an Ubuntu 18.04 system with an Intel Core i7-10750H and a NVIDIA GeForce RTX 2060 Mobile. Such models, as explained in Section III, are the starting point for the Vitis AI framework and are also considered as the software baselines. On the FPGA side, we exploit Vitis AI 1.4.1 and Vitis 2021.1 default image for the ZCU104. To validate SENECA, we select three metrics, namely throughput, efficiency, and accuracy.

1) Throughput and Efficiency: During the evaluation of SENECA, we calculate the throughput as Frames Per Second (FPS), in other words, how many images we can semantically segment in one second. On the other hand, we measure the DC power consumption in Watt during the inference phase with the Voltcraft 4000 energy logger for FPGAs and NVIDIA *smi* with the laptop plugged in for the GPU. Since throughput and power consumption are dependent variables, we exploit for the models' evaluation the Energy Efficiency (EE) as the ratio between the two as:

$$EE = \frac{FPS}{Watt} = \frac{FRAMES}{Joule} \tag{3}$$

2) Accuracy: We explore the semantic segmentation ability through the widely employed Dice Similarity Coefficient (DSC) defined as:

$$DSC = \frac{2\,|P \cap G|}{|P| + |G|} \tag{4}$$

where P is the predicted segmentation and G is the ground truth label. Further analysis of the networks' performance is based on the Recall (or True Positive Rate TPR) and Specificity (or True Negative Rate TNR). Such metrics help evaluate the network overreach towards detecting True Positives (TP) or avoiding False Positives (FP). The metrics are defined as:

$$TPR = \frac{|P \cap G|}{|G|} \tag{5}$$

$$TNR = \frac{|P^c \cap G^c|}{|G^c \cap P|} \tag{6}$$

where P^c and G^c are the complement sets of prediction and ground truth segmentation, respectively.

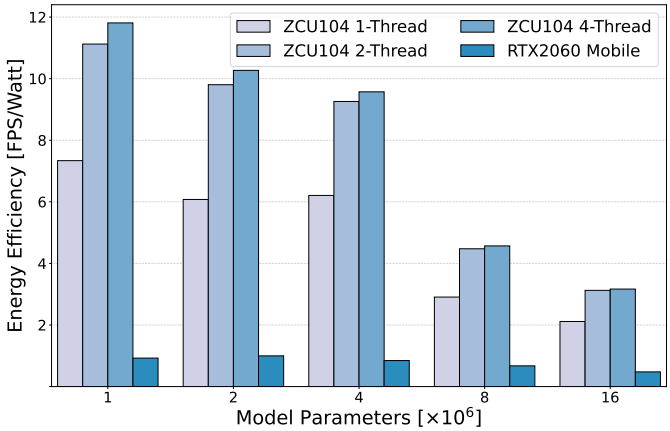

Fig. 3: Average energy efficiency for each model. The float models have been tested on a NVIDIA GeForce RTX 2060 Mobile. The quantized versions have been tested on the Xilinx ZCU104 evaluation board using 1, 2 and 4 threads.

B. Energy Efficiency Evaluation

As introduced in Section III, we develop five models with a variable number of trainable parameters, layers, and filters. To identify the best-performing model, we deploy all of them on FPGA and run the inference of 2000 images ten times with an increasing number of threads (1, 2, and 4). Since the accuracy does not depend on the number of threads employed and, therefore, is independent of both the power consumption and the throughput, we first select for each model (1M, 2M, 4M, 8M, and 16M) the best configuration in terms of energy efficiency and then we further consider the accuracy.

For these reasons, Figure 3 shows for each model the four considered configurations, namely the FP32 baseline GPU configuration and the three INT8 quantized FPGA versions with 1, 2, and 4 threads. We can see how the quantized configurations are always better than their GPU-based counterparts, with an increase in energy efficiency that varies between $12.76\times$ (1M) and $6.63\times$ (16M) when considering the four-thread configurations. In terms of Joule, for the same amount of processed frames, the FPGA configurations consume just between the 7.8% (1M) and 15.14% (16M) of the Joules used by their GPU-based counterparts. On the other hand, there is an energy efficiency improvement among the FPGA-based configurations when increasing the number of computing threads until 4. From a more in-depth analysis, we have seen that further addition of threads would not be

TABLE IV: FPS, Watt, EE, and DSC comparison between the FP32 model, evaluated on a NVIDIA GeForce RTX 2060 Mobile GPU, and the INT8 model evaluated on a Xilinx ZCU104 with 4 threads. Results are provided as $\mu \pm \sigma^2$ of 10 runs.

	FPS		Watt		Energy Efficiency $[\frac{FPS}{Watt}]$		DSC [%]	
Configuration	FP32	INT8	FP32	INT8	FP32	INT8	FP32	INT8
1M	72.20± 0.47	**335.40 ± 0.34**	78.01 ± 0.61	28.40 ± 0.02	0.93 ± 0.01	**11.81 ± 0.02**	92.98±0.16	93.04±0.07
2M	77.45 ± 0.14	254.87± 0.20	77.63 ± 0.91	**24.82 ± 0.02**	1.00 ± 0.01	10.27 ± 0.01	92.98±0.16	93.01±0.07
4M	65.90 ± 0.30	273.17± 0.21	77.94 ±0.54	28.54 ± 0.06	0.85 ± 0.01	9.57 ± 0.02	93.41±0.16	93.49±0.07
8M	52.22 ± 0.31	127.91 ± 0.06	77.56±0.90	28.00 ± 0.04	0.67 ± 0.01	4.57 ± 0.01	93.53±0.16	93.65±0.07
16M	37.23± 0.42	98.12 ± 0.19	77.99±0.97	30.98± 0.15	0.48 ± 0.01	3.17 ± 0.02	93.76±0.16	**93.84±0.07**

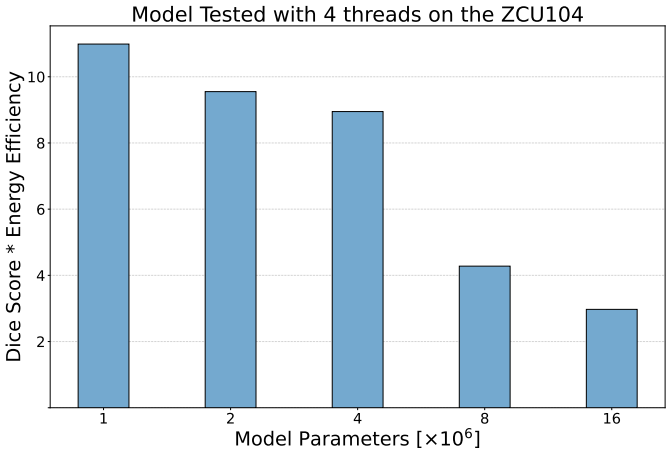

Fig. 4: Energy efficiency associated with models accuracy. Models tested with 4-threads on the Xilinx ZCU104.

beneficial in terms of energy efficiency; indeed, instantiating eight or more threads requires more power without a gain in FPS. We can ascribe this behavior to the overall run-time, which can time multiplex up to 4 threads without introducing too much overhead.

Figure 3 also highlights the decreasing trend of the energy efficiency associated with bigger models. As expected, a higher number of parameters requires more memory to be stored and computational resources, such as time and power. This trend is confirmed by Table IV, where we can see how bigger models segment fewer images per second. Indeed, for both GPU- and FPGA-based configurations, we can see how the 1M and 2M models reach higher FPS values, while the power necessary to segment the images seems less affected by the model dimensions.

Based on these analyses, we select as candidates to become the best model all the FPGA-based models running on 4 threads. We then explore them in terms of accuracy to finally identify the model to deploy within SENECA.

C. Accuracy Evaluation

From now on, when we refer to the 1M, 2M, 4M, 8M, and 16M, we are referring to their 4 threads configurations, so we no longer specify it. To select the best model, we should study the overall reached accuracy; indeed, we consider the DSC computed as the weighted mean of single organs DSCs. Figure 4 reports the results of the accuracy evaluation formulated as:

$$DSC_i \cdot EE_i \tag{7}$$

where DSC_i is the Dice Score of the model configuration *i*, while EE_i is the Energy Efficiency of the configuration i as in Equation (3).

Figure 4 shows a similar trend to Figure 3, where the smaller models perform better then the bigger ones. Indeed, we can see how the 1M model reaches a $3.7\times$ improvement compared to the 16M and a $1.15\times$ over the 2M one. Also, in this case, if we look at the contribution of the two components, we should notice from Table IV that the most impacting one is still the FPS rate. The DSC among all the five INT8 models is almost constant with a maximum variation of 0.83 percentage points that is negligible compared to the delta in FPS that reaches 237.28. Moreover, all the FPGA-based models reach comparable accuracy with their GPU counterparts, further supporting the decision to move the inference of DL models to FPGA-based devices, especially in computationally and energetically restraint scenarios.

Based on all the run experiments, we select the 1M FPGA-based model running on 4 threads as the best trade-off between energy consumption, FPS, and accuracy. Indeed, it reaches a $4.65\times$ FPS rate compared to its GPU counterpart and a $3.42\times$ on the 16M FPGA model, with a negligible accuracy loss. Therefore from now on, this model will be referred to as SENECA, and we will further explore its ability to semantically segment the various considered organs.

D. Best Model Accuracy Evaluation

To better explore the benefits of an embedded FPGA-based system for medical semantic segmentation, we have further studied the ability of SENECA to overcome the class imbalance problem both qualitatively and quantitatively.

As a first step, we have qualitatively evaluated the reached accuracy; indeed, Figure 5 shows some sample images where for each row we report the input image, the ground truth label from CT-ORG, the SENECA output, and the output of its GPU counterpart. As expected from the results in Table V, the visual inspection demonstrates the excellent capability of SENECA to deal with different organs.

We then move to a more systematic analysis of the accuracy of SENECA by exploring the performance by organs. Figure 6 shows the boxplots of the DSCs for each target organ. We

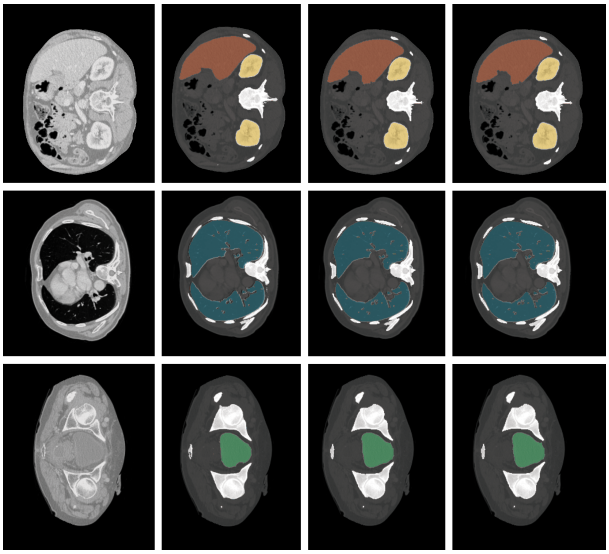

Fig. 5: Visual comparison of the results obtained by SENECA. From left to right: original CT slices, ground truth segmentations, INT8 SENECA segmentations, FP32 SENECA segmentations. The liver is shown in red, bladder in green, lungs in blue, kidneys in yellow and bones in white.

can see how they present a stable behavior around their mean value even when their biological shape is not, such as the bones. The reached DSC is higher than 90% for bigger organs while around 80% for the smaller ones. Thanks to their regular shape, high contrast, and dimensions, the lungs reach the best result. On the other hand, looking at the kidneys and bladder, the DSCs reflect the organ frequency in the dataset; indeed, the bladder reaches the lowest DSC being also the less represented one. From Figure 6 we can see that even if there is still a gap between small and big organs in terms of accuracy, our custom training loss reduce the impact of class imbalance; in fact, in the dataset, the lungs are $13.6\times$ more frequent than the bladder, but the lungs have just $1.21\times$ higher DSC.

Apart from the achieved low results variability, which demonstrates the robustness of the proposed solution, another strength resides in the TPR and the TNR. Indeed, SENECA reaches remarkable performance in detecting TPs with a global sensitivity of $93.06\%\pm0.07$, and a nearly perfect ability to prevent FP cases with a global TNR of $99.75\% \pm 0.07$. A deeper analysis can then conclude that the proposed network shows a more conservative behavior when detecting the organs' edges since the minimization of the number of FPs.

E. Comparison with baseline and literature

As the last step in the SENECA evaluation, we have compared it to its GPU counterpart and the work proposed along with the CT-ORG dataset [17]. We will not compare to works like [22], [23] since they only approach sub-tasks like the segmentation of smaller organs or adjacent ones, not having to deal with relevant class imbalance problems nor with a large number of organs simultaneously.

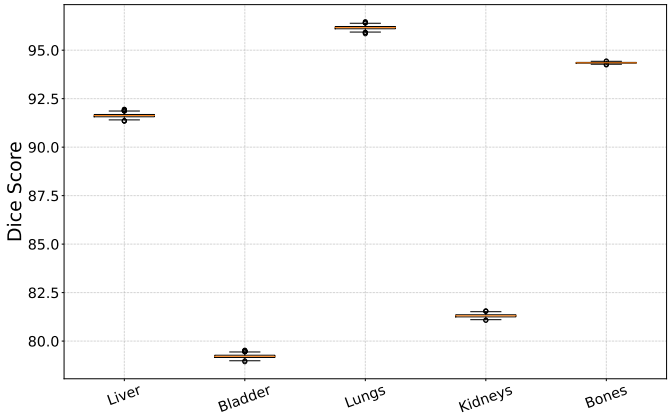

Fig. 6: Box plots of the dice score for all the organs segmented by SENECA on the CT-ORG dataset.

TABLE V: Comparison of SENECA between the FPGA 4-threads congifuration, the GPU configuration and the results obtained by [17]. Results are reported as $mean \pm std$.

	FPGA	GPU	CT-ORG [17]
FPS	**335.4 ± 0.34**	72.20±0.47	[17-197]†
Energy Efficiency	**11.81±0.02**	0.93±0.01	n/a
Global DSC	**93.04±0.07**	92.98±0.16	88.17±5.16
Liver DSC	91.63±0.09	91.01±0.20	**92.00±3.6**
Bladder DSC	79.21±0.09	**83.25±0.20**	58.10±22.3
Lungs DSC	**96.16±0.09**	95.93±0.21	93.80±5.9
Kidneys DSC	81.3±0.08	82.02±0.18	**88.20±7.9**
Bones DSC	94.35±0.03	**94.64±0.06**	82.70±7.6

† Value extracted from the data provided in [17]. The range is the reported mean execution time per patient divided by the maximum/minimum number of images composing a CT stack.

As explained in Section III-D, the Vitis AI Quantizer tunes the network weights based on the provided calibration dataset. For this reason, by looking at Table V, we can see that more frequent organs such as the liver and lungs show better DSCs after quantization while smaller organs, such as the bladder, can face some performance losses compared to the GPU implementation. On the other hand, given that the models' global DSCs are weighted on the frequency of the organs, we obtain slightly better global DSC results with the quantized version. Moreover, as previously introduced, SENECA delivers higher throughput and is more energy-efficient than its GPU version. It improves the FPS rate and the energy efficiency by $4.65\times$ and $12.7\times$, respectively. Indeed, the compiled quantized model is able to completely take advantage of hardware resources and design parallelism of the Xilinx ZCU104.

We have compared SENECA to the CT-ORG 3D U-Net [17], and Table V reports the obtained results. By comparing the DSC of the single organs, we can see how SENECA achieves remarkable results for bigger organs but also reaches good results on smaller organs such as bladder and kidneys, proving the ability of our weighted Focal Tversky Loss to deal with class imbalance. Compared to [17] we obtain similar results regarding liver and lungs. On the other hand, the FCN

proposed by [17] shows a higher DSC for kidneys but at the cost of a high variability represented by the 7.9 points of standard deviation. This issue is observable on all the predicted organs, particularly on the bladder where SENECA outperforms the 3D U-Net by over 20% with a considerable standard deviation reduction of more than 22%. This low variability achieved by the presented network shows the capability of our 2D U-Net to provide stable performance among different and variable organs. Moreover, SENECA shows an increment between $1.7\times$ and $19.73\times$ in terms of FPS compared to the CT-ORG 3D U-Net, which uses four GPUs (whose model is not specified).

V. Conclusions and Future Work

Within this work, we have proposed an open-source embedded quantized semantic segmentation network for medical images. Moreover, with SENECA, we have tailored semantic segmentation at the edge for a medical scenario along with its methodological workflow. Then, we have demonstrated with an extensive evaluation the benefits of employing an embedded reconfigurable hardware accelerator to improve the FPS rate of $4.65\times$ and the energy efficiency of about $12.7\times$ at inference time compared to GPU. Finally, we preserved the accuracy of the non-quantized model and improved the model stability compared to GPU and the CT-ORG work [17].

Future work - as future work, we will explore the possibility of further addressing the class imbalance problem, and we will study its generalizability more in-depth. Moreover, we will evaluate some pruning techniques to additionally improve throughput and energy efficiency.

Acknowledgment

The Authors would like to thank the Xilinx University Program (XUP) for the ZCU104 donation.

References

[1] Y.-J. Zhang, *Advances in image and video segmentation*. IRM Press, 2006.

[2] 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.

[3] 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*, 2022.

[4] 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.

[5] 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.

[6] L. Cai, J. Gao, and D. Zhao, "A review of the application of deep learning in medical image classification and segmentation," *Annals of translational medicine*, vol. 8, no. 11, 2020.

[7] X. Liu, L. Song, S. Liu, and Y. Zhang, "A review of deep-learning-based medical image segmentation methods," *Sustainability*, vol. 13, no. 3, p. 1224, 2021.

[8] S. P. Singh, L. Wang, S. Gupta, H. Goli, P. Padmanabhan, and B. Gulyás, "3d deep learning on medical images: a review," *Sensors*, vol. 20, no. 18, p. 5097, 2020.

[9] 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.

[10] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *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.

[11] 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.

[12] 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.

[13] J. Fowers, K. Ovtcharov, M. Papamichael, T. Massengill, M. Liu, D. Lo, S. Alkalay, M. Haselman, L. Adams, M. Ghandi, S. Heil, P. Patel, A. Sapek, G. Weisz, L. Woods, S. Lanka, S. Reinhardt, A. Caulfield, E. Chung, and D. Burger, "A configurable cloud-scale dnn processor for real-time ai," in *Proceedings of the 45th International Symposium on Computer Architecture, 2018*. ACM, June 2018. [Online]. Available: https://www.microsoft.com/en-us/research/publication/a-configurable-cloud-scale-dnn-processor-for-real-time-ai/

[14] P. D'Alberto, J. Ma, J. Li, Y. Hu, M. Bollavaram, and S. Fang, "Dpuv3int8: A compiler view to programmable fpga inference engines," *arXiv preprint arXiv:2110.04327*, 2021.

[15] K. Clark, B. Vendt, K. Smith, J. Freymann, J. Kirby, P. Koppel, S. Moore, S. Phillips, D. Maffitt, M. Pringle *et al.*, "The cancer imaging archive (tcia): maintaining and operating a public information repository," *Journal of digital imaging*, vol. 26, no. 6, pp. 1045–1057, 2013.

[16] 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

[17] 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.

[18] 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.

[19] S. Hooker, "The hardware lottery," *Communications of the ACM*, vol. 64, no. 12, pp. 58–65, 2021.

[20] L. Jiao, C. Luo, W. Cao, X. Zhou, and L. Wang, "Accelerating low bit-width convolutional neural networks with embedded fpga," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2017, pp. 1–4.

[21] E. Reggiani, E. Del Sozzo, D. Conficconi, G. Natale, C. Moroni, and M. D. Santambrogio, "Enhancing the scalability of multi-fpga stencil computations via highly optimized hdl components," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 14, no. 3, pp. 1–33, 2021.

[22] E. Gibson, F. Giganti, Y. Hu, E. Bonmati, S. Bandula, K. Gurusamy, B. Davidson, S. P. Pereira, M. J. Clarkson, and D. C. Barratt, "Automatic multi-organ segmentation on abdominal ct with dense v-networks," *IEEE transactions on medical imaging*, vol. 37, no. 8, pp. 1822–1834, 2018.

[23] Y. Wang, Y. Zhou, W. Shen, S. Park, E. K. Fishman, and A. L. Yuille, "Abdominal multi-organ segmentation with organ-attention networks and statistical fusion," *Medical image analysis*, vol. 55, pp. 88–102, 2019.

[24] 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.

[25] N. Abraham and N. M. Khan, "A novel focal tversky loss function with improved attention u-net for lesion segmentation," 2018.

[26] S. Jadon, "A survey of loss functions for semantic segmentation," in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE, 2020, pp. 1–7.

[27] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[28] X. Zhang, J. Wang, C. Zhu, Y. Lin, J. Xiong, W.-m. Hwu, and D. Chen, "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.

[29] I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, and D. Soudry, "Improving post training neural quantization: Layer-wise calibration and integer programming," *ArXiv*, vol. abs/2006.10518, 2020.