POLITECNICO DI TORINO
Repository ISTITUZIONALE

HW-SW Optimization of DNNs for Privacy-Preserving People Counting on Low-Resolution Infrared Arrays

Original
HW-SW Optimization of DNNs for Privacy-Preserving People Counting on Low-Resolution Infrared Arrays / Risso, M., Xie, C., Daghero, F., Burrello, A., Mollaei, S., Castellano, M., Macii, E., Poncino, M., JAHIER PAGLIARI, D.. - ELETTRONICO. - (2024), pp. 1-6. (Design, Automation and Test in Europe Conference and Exhibition, DATE 2024 Valencia (ESP) 25-27 March 2024).

Availability:
This version is available at: 11583/2991616 since: 2024-08-09T07:24:54Z

Publisher:
Institute of Electrical and Electronics Engineers

Published
DOI:

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

01 November 2026

This paper has been accepted for publication in the DATE 2024 conference
©2024 IEEE.

HW-SW Optimization of DNNs for Privacy-preserving People Counting on Low-resolution Infrared Arrays

Matteo Risso*, Chen Xie*, Francesco Daghero*, Alessio Burrello*, Seyedmorteza Mollaei*,
Marco Castellano†, Enrico Macii*, Massimo Poncino*, Daniele Jahier Pagliari*
* Politecnico di Torino, Turin, 10129, Italy † ST Microelectronics S.r.l., Cornaredo, 20010, Italy
Emails: name.surname@polito.it, name.surname@st.it

arXiv:2402.01226v1 [cs.LG] 2 Feb 2024

***Abstract*—Low-resolution infrared (IR) array sensors enable people counting applications such as monitoring the occupancy of spaces and people flows while preserving privacy and minimizing energy consumption. Deep Neural Networks (DNNs) have been shown to be well-suited to process these sensor data in an accurate and efficient manner. Nevertheless, the space of DNNs' architectures is huge and its manual exploration is burdensome and often leads to sub-optimal solutions. To overcome this problem, in this work, we propose a highly automated full-stack optimization flow for DNNs that goes from neural architecture search, mixed-precision quantization, and post-processing, down to the realization of a new smart sensor prototype, including a Microcontroller with a customized instruction set. Integrating these cross-layer optimizations, we obtain a large set of Pareto-optimal solutions in the 3D-space of energy, memory, and accuracy. Deploying such solutions on our hardware platform, we improve the state-of-the-art achieving up to 4.2× model size reduction, 23.8× code size reduction, and 15.38× energy reduction at iso-accuracy.**

***Index Terms*—Deep Learning, Neural Architecture Search, TinyML, MCUs, Smart Sensors**

I. Introduction

In the age of pervasive computing, precise counting of people in public and private locations is critical in sectors such as smart buildings and cities, to monitor occupancy and people flows [1].

Classic people counting solutions rely on acquiring a video stream and processing it with a Deep Neural Network (DNN) [2], [3]. However, preserving the privacy of individuals is of utmost importance for these applications, and traditional video-based counting systems fail to address this problem by gathering high-resolution visual data [2], [3], resulting in ethical and legal concerns, especially for public spaces. To address this issue, many researchers have proposed the use of low-resolution infrared (IR) sensors [1], [4], that capture body heat patterns rather than visual details, allowing for an unobtrusive and anonymous people counting that safeguards sensitive information.

Aside from privacy, achieving low energy consumption is another key objective for people counting systems, e.g., in public areas with no access to the electrical grid. To this end, in-sensor computing is advantageous, as it reduces the dependence on energy-hungry data transfers over a wireless network [5]. Performing DNN inference on a sensor node, however, introduces a new set of challenges. The complex nature of deep learning models requires intensive computations that may strain the limited capabilities of edge hardware in terms of processing and memory.

This work has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 101007321. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Belgium, Czech Republic, Germany, Italy, Sweden, Switzerland, Turkey.

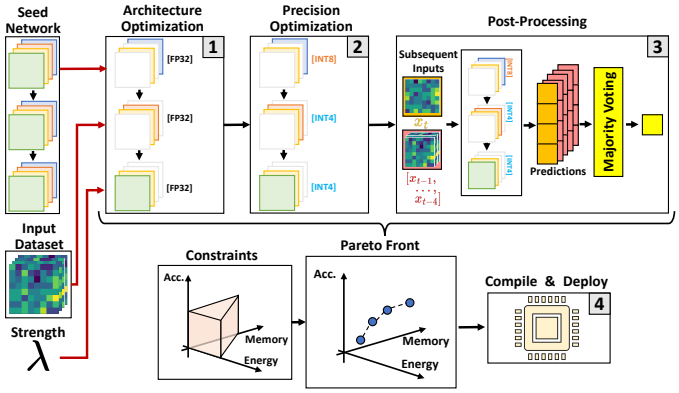


Fig. 1. Overview of the full-stack optimization flow.

In [4], an extensive evaluation of different efficient DNNs led to a rich collection of Pareto-optimal solutions for people-counting on low-resolution IR sensors on the LINAIGE [6] dataset. However, a crucial limitation of [4] lies in its hand-tuned selection of model architecture configurations. Relying entirely on human expertise may lead to sub-optimal solutions, biased by "rules-of-thumb" and designer intuitions. Furthermore, this approach is severely time-consuming, forcing designers to consider only coarse-grain search spaces. The result is a highly inefficient exploration with a poor ratio between Pareto-optimal solutions and the total number of explored architectures (e.g., 0.8% in [4]).

To address these limitations and improve the State-of-the-Art (SotA) for in-sensor IR-based people counting, this work introduces a novel optimization flow that automates the exploration of DNN architectures and quantization formats, along with a novel smart sensor prototype, called MAUPITI, built extending the IBEX [7] RISC-V core to efficiently support low-precision vector operations (Sec. III-B2). Fig. 1 summarizes the optimization flow. Its main components are:

- Complexity-aware Differentiable Neural Architecture Search (DNAS) [8] to automatically identify good settings of key DNN hyper-parameters (Sec. III-A1).
- Mixed-precision quantization of weights and activations of Pareto-optimal DNNs using `INT4` and `INT8` data formats (Sec. III-A2).

©2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

- A post-processing technique that improves accuracy exploiting the temporal correlation of subsequent inputs with negligible memory and energy overheads (Sec. III-A3).
- Compilation and lightweight inference runtime support for the custom ISA extensions of MAUPITI (Sec. III-B3).

Thanks to the proposed flow, we were able to obtain a rich collection of Pareto-optimal solutions spanning up to one order of magnitude in memory footprint. When compared to SotA, we achieve up to 4.2× reduction in memory footprint, 23.8× in code size, and 15.38× in energy at iso-accuracy.

II. Background & Related Work

A. Neural Architecture Search (NAS)

Nowadays, NAS tools represent the go-to solution to help designers in the initial stages of DNN optimization. These tools automatically search for the best DNN architectures among a large number of alternatives described as combinations of different layers and/or hyper-parameters. Noteworthy, modern NAS can consider task-specific performance metrics (e.g., accuracy) and non-functional metrics such as memory usage, latency, and energy [9].

In particular, Differentiable NAS (DNAS) methods optimize DNNs *while training them*, using gradient descent to solve a relaxed version of the architecture selection problem. In this way, they significantly reduce the search time with respect to earlier iterative approaches based on Reinforcement Learning or Evolutionary Algorithms (which required 1000s of GPU hours, e.g. [10]), making it comparable to a single training [9], [11]. Some DNAS implementations construct *supernets*, i.e., DNNs with multiple alternative paths, and use the training optimizer to assign a higher probability of being selected to paths that obtain the best accuracy vs cost trade-offs [9], [11]. However, supernets are still very costly to train in terms of GPU memory and latency. Mask-based DNAS [8], [12] further reduce optimization costs by exploring a search-space defined by *sub-architectures* contained within a standard DNN, known as the *seed*. Alternative architectures are built by subtraction, removing parts of each layer, such as some channels in a convolution. In practice, these sub-networks are emulated during training by selectively pruning portions of the seed via *trainable masks*. While mask-based DNAS strategies are limited to generating networks stemming from the seed, they foster a much more lightweight and fine-grained exploration of the search space [8].

B. Quantization and Mixed-Precision

Integer quantization is a crucial DNN optimization that replaces floating-point weights and activations with low-bitwidth integers. Quantization leads to enhancements in model size, speed, and energy efficiency [13]. Furthermore, replacing floating-point computations with integer ones enables the execution of DNNs even on hardware without a Floating Point Unit (FPU). While quantization can also be applied post-training, simulating its effect with the so-called Quantization-Aware Training (QAT) [13], is helpful to keep higher accuracy.

Traditional *fixed-precision* quantization employs a uniform bit-width N_b (typically 8 bits) across the model. Recently, however, mixed-precision DNNs using different bit-widths for different portions of the network [14], [15] have been shown to provide added benefits in terms of time, memory, and energy, particularly when the underlying hardware supports native sub-byte operations, as in the case of MAUPITI.

C. Privacy-Preserving People Counting

People counting is important in applications such as smart homes, public security, and pedestrian flow analysis [1], [16]. *Instrumented* people counting solutions leverage the Bluetooth or Wi-Fi signals coming from user devices such as wearables and smartphones [17]. However, their applicability is limited by the strict dependency on individuals' voluntary participation. *Uninstrumented* solutions, on the other hand, rely on external sensors, such as optical cameras, thermopiles, IR arrays, etc. [18]. Vision-based approaches obtain remarkable results [2], but introduce new important privacy concerns related to storing and processing images that include private user information, such as facial details. Low-resolution IR sensors, instead, can detect people without privacy concerns, since they acquire low-resolution thermal images (e.g., 8x8 or 16x16) while allowing much higher counting accuracy with respect to, e.g., single passive IR sensors [19].

Previous works that exploit IR arrays for people counting can be divided into deterministic (non-data-driven) approaches [1], [19], [20] and Machine Learning (ML)-based methods [4], [21]–[23]. Deterministic algorithms are based on hand-crafted feature-extraction procedures tailored for specific application scenarios. In general, all these solutions suffer from limited generality to new environments. For instance, [20] only counts people entering/exiting a room using doorway-mounted IR arrays. Instead, [1], [19] monitor more general environments with ceiling/side wall-mounted IR arrays. However, [1] relies on a network of multiple IR sensors, while [19] is based on a single low-resolution IR array, but suffers from low accuracy as analyzed in [4].

Among ML-based solutions, [21] analyzed multiple classic models such as Random Forests (RFs), Support Vector Machines (SVMs) etc., showing that RFs achieve the highest accuracy on low-resolution (8x8) arrays, but also that results are strongly dependent on the selected hand-crafted feature set. DNNs are, therefore, a promising way to get rid of manual feature extraction while achieving even higher accuracy. [22], [23] apply different types of DNNs, including Convolutional Neural Networks (CNNs), Feedforward Neural Networks (FNNs), or Gated Recurrent Units (GRUs) to this task. While proving the superior accuracy of deep learning, these solutions are deployed only on high-performance, energy-hungry, PC-class processors [21], [22], and/or utilize relatively high-resolution arrays (e.g., 80x60), which reduces privacy protection. More recently, [4] provided an extensive comparative analysis of 6 efficient DNN model families for people counting based on a single ultra-low-resolution (8x8) IR array, deploying the results on a commercial Microcontroller (MCU). However, the architectural exploration in [4] was manual and coarse-grained, limiting the possible Pareto-optimal solutions that could be found. Moreover, the MCU considered in [4] did not include specific hardware features to accelerate the execution of low-precision DNNs, which may lead to further efficiency benefits.

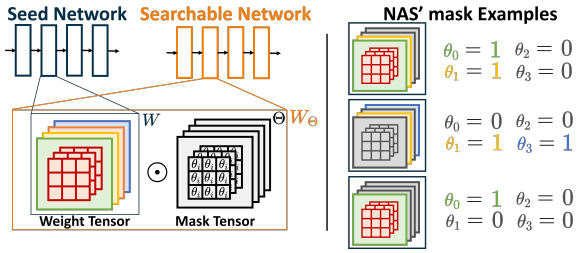


Fig. 2. Left: The PIT mask-based DNAS scheme. Right: three example results from the search procedure on a layer with four filters.

III. Methods

This section details our full-stack optimization flow and our novel HW platform, comprising a low-power IR sensor and a RISC-V core optimized for low-precision DNN inference.

The goal of the flow, depicted in Fig. 1, is to obtain a rich set of DNN models for IR-based people counting, offering diverse trade-offs in terms of task performance and HW cost. The latter is measured in terms of n. of parameters (a proxy for memory), or n. of Multiply and Accumulate (MAC) operations (a proxy for energy). The three inputs to the flow are the training dataset, a seed DNN that acts as "blueprint" to generate all output solutions, and a scalar parameter λ used to control the trade-off between task performance and HW-cost. Note that the novelty of our method does not reside in the single steps, but in their concatenation into a full-stack flow and application to the IR-based people counting task.

A. Software Optimization Flow

1) Architecture Optimization: The architecture optimization step is based on the PIT [8] mask-based DNAS, as implemented in [24]. PIT starts from a seed CNN and explores sub-architectures contained in it by structurally pruning the output channels/features of convolutional and linear layers. We selected a mask-based approach due to its lower memory and time overhead compared to other NAS tools [9]–[11]. Moreover, such an approach allows to leverage SotA seeds as starting points for the exploration. In particular, we use the DNNs found manually in [4] as a starting point, demonstrating how fine-grained exploration results in new optimization opportunities.

As schematized in Fig. 2, PIT considers all convolutional/linear layers of the seed and couples, through Hadamard product $\odot$, each output channel c of the weight tensor W with a binarized trainable mask parameters θ_c:

$$W^c_\Theta = W^c \odot \mathcal{H}(\theta_c) \tag{1}$$

where $\mathcal{H}$ is the Heavised step function that binarizes θ and W^c denotes a specific slice of the weight tensor on the output channel dimension.

The obtained network is then inserted in a standard training loop where both weights W and masks θ are trained to minimize the following objective function:

$$\min_{W,\theta} \mathcal{L}(W;\theta) + \lambda \mathcal{C}(\theta) \tag{2}$$

In (2) $\mathcal{L}$ is a standard task-specific loss function (e.g., cross-entropy) and $\mathcal{C}$ is a differentiable model of a HW cost metric, e.g., the memory-footprint or the number of MACs. λ is a strength parameter that controls the balance between task performance and cost. The higher the value of λ, the higher the importance given to the minimization of the HW-related cost. Each value of λ will correspond to a specific architecture in the task-performance vs. cost space as detailed in Sec. IV-B.

2) Precision Optimization: Starting from the optimized architectures obtained with the NAS, we then explore DNN quantization by decreasing the precision of both data and operations from the standard `FLOAT32` to integer. We first fold Batch-Normalization (BN) operations with previous convolutional/linear layers, to reduce the total number of operations. Then, each floating point tensor T (both activations and weights) is quantized to an integer precision N_b, by means of an *affine transformation*:

$$\hat{T} = \text{round}\left(\frac{T-\alpha}{\beta-\alpha}(2^{N_b}-1)\right) \tag{3}$$

where $\hat{T}$ is the integer image of T and $[\alpha, \beta]$ can either represent the extremes of the variation range for T or can be learned during training [25]. In this work, we consider a range-based quantization for weights and a learnable one for activations. Moreover, we apply QAT, which is particularly important to recover part of the floating-point accuracy when BN folding and sub-byte quantization are employed.

Our preliminary QAT experiments showed a too high accuracy drop for precisions lower than `INT4` precision. This led to the design choice of supporting only `INT4` and `INT8` precisions in the MAUPITI hardware (see Sec. III-B2). We use a mixed-precision scheme [14] with a layer-wise granularity to explore different precision assignments, targeting the aforementioned formats. Our method differs with respect to standard mixed-precision, which independently assigns a bit-width to weights and activations. In fact, in order to minimize HW overheads, MAUPITI only supports 4x4-bit and 8x8-bit vectorial MAC operations, i.e., the precision assignment can be different for different layers, but must be the same for weights and activations of the same layer. Given the limited search space (detailed in Sec. IV), we run a complete exploration of all the possible alternatives using the QAT capabilities of [24].

3) Post-Processing: The third step of the flow of Fig. 1 consists of a simple yet effective post-processing technique based on majority voting, i.e., *mode inference*. The rationale is to take advantage of multiple classification results to generate final predictions with lower variance, by exploiting the temporal correlations among subsequent frames. In practice, the same DNN classifier is applied independently to each input. Then, the final output is built as the most frequently predicted class over a sliding window of recent frames, thus filtering out sporadic mispredictions. This solution is particularly interesting from the edge computing perspective, having approximately the same memory cost as a single-frame classifier. A similar approach was considered in [4], but in that paper, the network was executed multiple times, discarding previous predictions, thus incurring a large latency and energy cost. In our post-processing, we avoid any re-computation by simply storing previous predictions in a FIFO data structure, thus bringing latency and energy overheads close to 0 too.

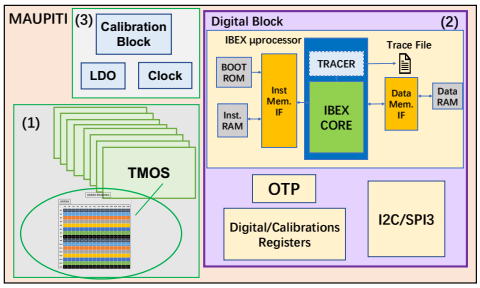


Fig. 3. The complete MAUPITI System.

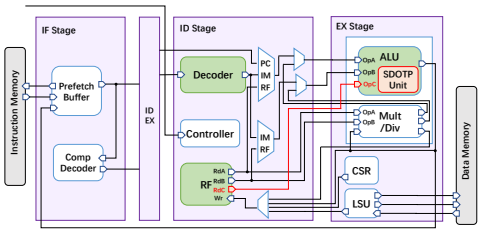


Fig. 4. Customized IBEX RISC-V core.

B. The MAUPITI Platform

1) Complete System: Fig. 3 depicts the complete MAUPITI system, which has been taped out in 130nm CMOS technology, is clocked at 20MHz, and can capture thermal images with a frame rate of 10 Frames Per Second (FPS). The chip includes three main components. A 16x16 array of thermal MOSFET (TMOS) sensors (1), sensitive to the heating effects of infrared radiation, with 8 parallel analog front-end processing chains, each able to acquire one row of pixels, thus enabling a two-step acquisition of a complete frame. Each TMOS draws $\approx 1\mu A@2.4V$, resulting in a total consumption of 0.62mW for the array. A digital processing block (2) which contains a customized IBEX core (highlighted in green) with its instruction and data memory (16KB each), and the corresponding interfaces, plus a boot ROM. Moreover, the block also includes an 80B One-Time Programmable Memory (OTP), an instruction tracer, calibrations registers for the TMOS sensors, and I2C and SPI3 communication Interfaces. Overall, the digital block consumes ≈0.9mW of power in FF conditions. Lastly, the system includes standard circuitry blocks (3) for clock and reset management, voltage regulation, etc.

2) Core Customization: The IBEX core has been customized to support quantized DNNs using `INT4` and `INT8` data formats, with the addition of an efficient arithmetic unit supporting low bit-width integer Single-Instruction-Multiple-Data (SIMD) Sum of Dot Product (SDOTP) operations. Figure 4 depicts the integration of the SDOTP unit into the IBEX pipeline. Hardware blocks highlighted in green have been modified, while the orange one is entirely new. In detail, the SDOTP unit performs a MAC operation between two 32-bit registers (RS1 and RS2), interpreting their content either as four 8-bit or as eight 4-bit *signed* values, and an additional 32-bit register (RD) is used as input and output of the accumulation. To achieve single-cycle latency, we implement the dot product using four independent 8-bit multipliers, and eight 4-bit multipliers, followed by an adder tree to sum up the partial products and the additional 32-bit operand. Replicating the multipliers for the two bitwidths, instead of sharing them, moves the new block out of the core's critical path, at the cost of an acceptable area increase [26]. The other key hardware modifications are in the Decoder, extended to support the new opcodes, and in the Register File (RF). The SDOTP uses RD both as source and destination, while the vanilla IBEX ALU only supports two input operands. Thus, we add a third input OpC to the ALU, which always takes data directly from the register file, and the MAC result is then written onto the same register. Accordingly, we must add a read port RdC to the RF, as shown in Figure 4.

With respect to [26], a SotA mixed-precision ISA extension for RISC-V MCUs, our solution focuses much more on containing area (i.e., cost) overheads, sacrificing some performance and flexibility, since we target a low-cost smart sensors application. To this end, we do not implement unsigned variants of the SDOTP, nor 8x4-bit or 4x8-bit versions, to support different precisions for weights and activations. Similarly, we do not implement 2-bit SDOTP since our QAT experiments showed that 2-bit precision causes strong accuracy degradations. We only support register addressing mode for the source operands (thus requiring separate loads before a dot product), and we do not implement combined MAC&Load. Lastly, we also avoid having separate instructions for simple dot product (DOTP), since those can be reduced to SDOTP by setting the third input register RD to 0. All these simplifications significantly reduce our area overhead to less than 7% w.r.t. the vanilla IBEX core. In contrast, [26] reported a 17.5% core area increase despite starting from a more complex baseline core.

3) Deployment Toolchain: To fully exploit the customized core, we added support for the new SDOTP instructions to the Gnu Compiler Collection (GCC) toolchain for RISC-V targets with the instruction set riscv32-imc. We then developed a minimal set of optimized kernels to implement the required DNN layers, largely inspired from [27], the SotA library for mixed-precision deep learning on RISC-V MCUs. In particular, we developed convolutional kernels taking as input weights/activations at 4/8 bit and writing their re-quantized output again on 4/8 bit. This enables a seamless integration with the quantized architectures found with the flow of Sec. III-A. Aside from convolution, we also have kernels for 2D max-pooling, whereas we avoid a dedicated implementation of linear (fully-connected) layers by re-using the convolutional kernels in the corner case of a 1x1 filter and 1x1 input/output feature maps, thus minimizing the code size of our runtime.

IV. Experimental Results

A. Setup

We test our flow on the open-source LINAIGE [6] dataset, which specifically targets the people counting task using 8×8 IR array sensor data. We use this dataset as it is the largest publicly available one, although the resolution is lower than that of the MAUPITI sensor. Collecting our own dataset will be part of our future work. The dataset contains 25110 labeled samples, split into 5 sessions (collected in different environments). Each sample is labeled with the number of people in the field of view, ranging from 0 to 3. In all experiments, the dataset is employed with the same training hyper-parameters and by following the *leave-one-session-out* cross-validation (CV) scheme described

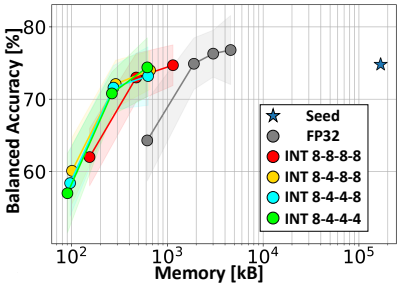


Fig. 5. Architecture and Precision Search Space exploration results. Different colors encode the different precisions' configurations.

in [4] with Session 1 (the largest) always kept in the training set and Sessions 2, 3, 4, 5 rotated as test-sets. Namely, we train for 500 epochs using the Adam Optimizer to optimize a cross-entropy loss, with a learning rate of 0.001 and a batch size of 128. We apply NAS and QAT only on Session 1 data, then apply QAT and fine-tune the found models on the training fold (which includes all sessions but one), and test on the left-out session. Models' performance is evaluated through the average of recall or Balanced Accuracy Score (BAS). Differently from [4], which runs the experiment only one time, we repeated them ten times with different random seeds. Each result is reported with mean and variance, thus providing a better statistical characterization.

As seed for the DNAS, we use the largest CNN configuration considered in [4], which includes a feature-extractor composed of two convolutional layers both with a 3×3 kernel, a stride of 1, and 64 output channels with a max-pooling layer in between. The network is concluded by two linear layers respectively with 64 and 4 output features. The two convolutions are followed by BN, and all layers except the output one use ReLU as non-linearity. Taking the largest configuration of [4], we aim to demonstrate that with our flow, we are able to match or improve the SotA thanks to a finer-grain, automated search.

All the code is written using Python 3.9, PyTorch 1.13.1 and employs the PLiNIO [24] library to implement NAS and QAT.

B. Architecture and Precision Space Exploration

The input of our flow is highlighted with a blue star in Fig. 5 in a BAS vs memory plane. Starting from such point and by applying the PIT [8] DNAS with different values of strength λ and using the number of parameters as cost $\mathcal{C}$ (See Eq. 2), we obtain the grey Pareto front. Noteworthy, when compared to the seed, we achieve up to 89× memory and 26.7× N. of MACs reduction and iso-BAS.

The next step of the flow is the mixed-precision quantization of the `FLOAT32` results obtained with PIT. The results are summarized in Fig. 5 with colored circles, where each color encodes a specific precision combination for the four considered layers. We only report solutions with the first layer quantized at 8-bit because using `INT4` to quantize the inputs led to severe accuracy degradations and resulted in sub-optimal solutions. Moreover, the "INT 8-8-4-8" is not reported because it is never part of the overall Pareto frontier. Thanks to quantization, we improve the `FLOAT32` front up to 2.3× in terms of memory while also improving the BAS up to 6.5% w.r.t. the left-most point of the grey curve. Overall, the memory and MACs reduction w.r.t. the seed reach 147× and 234× at iso-BAS.

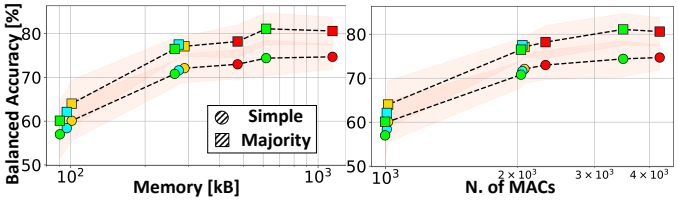


Fig. 6. Comparison of Pareto frontiers with and without post-processing.

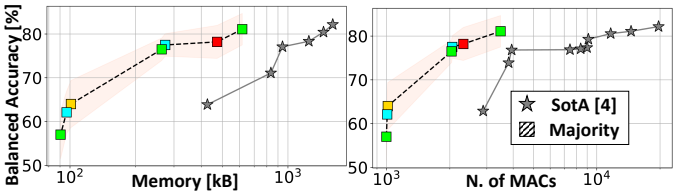


Fig. 7. State-of-the-Art Comparison.

C. Post-Processing Results

Fig. 6 shows the results of applying the proposed post-processing scheme on the best networks obtained with architecture and precision explorations. The circles correspond to the global Pareto fronts obtained from the NAS and QAT phases. For instance, the circles' curve in the leftmost plot is obtained merging the optimal points from all quantized curves in Fig. 5. The squares are the results of applying post-processing to the outputs of those DNNs. The left and right plots show how the majority-voting technique represents a plug-and-play strategy to improve the Pareto-frontier in the BAS vs. Memory and BAS vs. N. of MACs spaces. We considered a sliding window composed of the predictions on 5 subsequent frames, which demonstrated to be the most effective one on the dataset. Majority voting introduces a negligible delay in detecting people count changes equal to half of the window length (assuming all-correct predictions). In exchange, it achieves up to 6.7% BAS improvement at iso-memory and iso-MACs.

D. State-of-the-Art Comparison

Fig. 7 compares the results obtained with our proposed pipeline, and the SotA results presented in [4]. On the left-most plot, we compare the results in the BAS vs. Memory space, and on the right, in the BAS vs N. of MACs space. As shown, [4] achieved a slightly higher maximum BAS (+0.9%), which, however, is well within the standard deviation range of our most accurate model. On the other hand, when targeting a BAS higher than 80%, we obtain models up to 2.4× smaller and requiring 3.3× fewer MACs with respect to the SotA. Similarly, when comparing against the most memory-efficient DNN of [4], and the one with the lowest number of MACs (the left extremes of the two grey curves), our flow produces models that are respectively 4.2× smaller and require 2.9× fewer MACs for the same BAS.

E. Embedded Deployment Results

Table I reports the deployment results of the top scoring model (*Top*), the smallest model in terms of memory with a maximum drop in accuracy smaller than 5% (*-5%*), and the

TABLE I
DEPLOYMENT RESULTS.

Model	Platform	Code [B]	Data [B]	Energy [μJ]
Top	STM32	22840	10420	9.381
	IBEX	3476	1104	6.003
	MAUPITI	4152	1104	4.927
- 5%	STM32	22970	9060	6.854
	IBEX	3384	648	5.005
	MAUPITI	4052	648	4.525
Mini	STM32	22950	8410	5.640
	IBEX	2700	416	4.342
	MAUPITI	3208	416	4.067

smallest overall (*Mini*). Noteworthy, in our case, the latter corresponds also to the models with the lowest MACs. For the aforementioned architectures, we report the code size (Code), the memory occupation (Data), and the energy consumption per inference on three different platforms. Namely, we compare the proposed MAUPITI platform with an unmodified IBEX core, using no custom instructions. For these two targets, we use our deployment toolchain with identical compilation flags. Moreover, we compare MAUPITI to an off-the-shelf MCU solution, i.e., an STM32L4R5 core with models deployed on 8bits only using the proprietary X-CUBE-AI toolchain, which does not support mixed-precision. When comparing with IBEX, the *Top* model, achieves the largest gains in terms of energy. MAUPITI, at the cost of 7% area overhead, and despite a 2.2% post-synthesis power overhead, achieves up to 17.9% energy reduction. Note that the efficiency benefits of the low-precision SIMD in MAUPITI are limited by the small geometry of the considered layers, namely the small number of output channels, and would be higher for larger DNNs.

Concerning memory, we see a code size increase compared to the vanilla IBEX core. We have the largest increase (676 B) with the *Top* model. The reason is the more complex logic of MAUPITI kernels, with the baseline versions repeating a regular read-and-multiply pattern independently from the layer hyper-parameters. On the other hand, SIMD instructions read and perform operations on chunks of inputs, thus having to manage non-multiple-of-4(8) dimensions as "leftovers".

Thanks to our lightweight runtime, when comparing with STM32 solutions, MAUPITI shows up to $6.78\times$/$20.22\times$ code-size and data reduction. Even the most accurate network can easily fit the 16 kB of code memory and 16 kB of data memory available on the chip. STM32 executes DNNs up to $9\times$ faster than MAUPITI. This is partly due to the higher frequency of the core (120MHz versus the 20 MHz of MAUPITI), partly to the different ISA, and partly to the optimizations included in X-CUBE-AI, such as max-pooling fusion. Nevertheless, such reduction in latency is paid with a huge increase of power consumption of $13.2\times$, which makes MAUPITI up to 1.4x-1.9x more efficient in terms of energy.

Finally, when comparing with the deployment results of [4] we note that their smallest solution (leftmost grey star in Fig. 7) was reported to have a code size $23.8\times$ higher and to consume $15.38\times$ more energy compared to our equally accurate DNN (third square from the left in Fig. 7). Conversely, when comparing our *Top* model with the one in [4] (rightmost grey star in Fig. 7) we achieve $69\times$ code size reduction and $24.4\times$ energy reduction with a small BAS drop of 0.9%.

V. CONCLUSIONS

The yet accurate yet privacy-preserving monitoring of people flows in smart buildings and cities contexts represents key requirements. This work effectively proposes a full-stack optimization pipeline, from software down to hardware, that enables such application to use low-resolution IR arrays and perform DNN inference at the edge. The proposed flow is able to improve the SotA with up to $4.2\times$ memory reduction, $23.8\times$ code-size reduction, and $15.38\times$ energy reduction at iso-accuracy.

REFERENCES

[1] R. Rabiee *et al.*, "Multi-bernoulli tracking approach for occupancy monitoring of smart buildings using low-resolution infrared sensor array," *Remote Sensing*, 2021.

[2] S. Basalamah *et al.*, "Scale driven convolutional neural network model for people counting and localization in crowd scenes," *IEEE Access*, 2019.

[3] V. Nogueira *et al.*, "Retailnet: A deep learning approach for people counting and hot spots detection in retail stores," in *SIBGRAPI*, 2019.

[4] C. Xie *et al.*, "Efficient deep learning models for privacy-preserving people counting on low-resolution infrared arrays," *IEEE IoT J.*, 2023.

[5] J. Chen *et al.*, "Deep learning with edge computing: A review," *Proceedings of the IEEE*, 2019.

[6] C. Xie *et al.*, "Linaige," 2022. [Online]. Available: https://www.kaggle.com/dsv/3073276

[7] "Ibex." [Online]. Available: https://github.com/lowrisc/ibex

[8] M. Risso *et al.*, "Lightweight neural architecture search for temporal convolutional networks at the edge," *IEEE Trans. Comp.*, 2023.

[9] H. Cai *et al.*, "Proxylessnas: Direct neural architecture search on target task and hardware," *arXiv preprint arXiv:1812.00332*, 2018.

[10] M. Tan *et al.*, "Mnasnet: Platform-aware neural architecture search for mobile," in *IEEE/CVF CVPR*, 2019.

[11] H. Liu *et al.*, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[12] A. Wan *et al.*, "Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions," in *IEEE/CVF CVPR*, 2020.

[13] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *IEEE/CVF CVPR*, 2018.

[14] Z. Cai *et al.*, "Rethinking differentiable search for mixed-precision neural networks," in *IEEE/CVF CVPR*, 2020.

[15] M. Risso *et al.*, "Channel-wise mixed-precision assignment for dnn inference on constrained edge nodes," in *IEEE IGSC*, 2022.

[16] A. Giaretta *et al.*, "On the people counting problem in smart homes: Undirected graphs and theoretical lower-bounds," *J. Ambient Intell. Humaniz. Comput.*, 2021.

[17] W. Xi *et al.*, "Electronic frog eye: Counting crowd using wifi," in *IEEE INFOCOM*, 2014.

[18] K. Hashimoto *et al.*, "People count system using multi-sensing application," in *Transducers' 97*. IEEE, 1997.

[19] P. Industry, "Grid-eye application note on social distancing. people detection and tracking with ceiling mounted sensors," 2020.

[20] C. Perra *et al.*, "Monitoring indoor people presence in buildings using low-cost infrared sensor array in doorways," *Sensors*, 2021.

[21] V. Chidurala *et al.*, "Occupancy estimation using thermal imaging sensors and machine learning algorithms," *IEEE Sensors J.*, 2021.

[22] M. Bouazizi *et al.*, "Low-resolution infrared array sensor for counting and localizing people indoors: When low end technology meets cutting edge deep learning techniques," *Information*, 2022.

[23] M. Kraft *et al.*, "Low-cost thermal camera-based counting occupancy meter facilitating energy saving in smart buildings," *Energies*, 2021.

[24] D. J. Pagliari *et al.*, "Plinio: A user-friendly library of gradient-based methods for complexity-aware dnn optimization," 2023.

[25] J. Choi *et al.*, "Pact: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.

[26] A. Garofalo *et al.*, "Xpulpnn: Enabling energy efficient and flexible inference of quantized neural networks on risc-v based iot end nodes," *IEEE TETC*, 2021.

[27] N. Bruschi *et al.*, "Enabling mixed-precision quantized neural networks in extreme-edge devices," in *ACM CF*, 2020.