A 772µJ/frame ImageNet Feature Extractor Accelerator on HD Images at 30FPS

Ivan Miro-Panades*[1], Vincent Lorrain*[2], Lilian Billod[1], Inna Kucher[2], Vincent Templier[2], Sylvain Choisnet[1], Nermine Ali[2], Baptiste Rossigneux[2], Olivier Bichler[2], Alexandre Valentian[1]
[1]Univ. Grenoble Alpes, CEA, List, Grenoble, France; [2]Univ. Paris-Saclay, CEA, List, Palaiseau, France;
*Equally Credited Authors (ECAs); Email: ivan.miro-panades@cea.fr

***Abstract*— Many applications benefit from AI inference at the edge, in industry, agriculture and transportation domains. The observed trend in image/video analysis is to increase the resolution of sensors, thus increasing the need for powerful, yet ultra-low power and low-latency hardware solutions. In this paper, we explore a novel approach to meet these conflicting requirements and we propose a whole new class of accelerator: the Feature Extractor Accelerator (FEA). This solution enables processing up to HD images (1280x720) at 30 frames per second (FPS), using a fraction of the energy of the actual image sensor: it consumes at most 23.2mW, with a 6.9ms latency, while reaching 70.42% accuracy on ImageNet. This approach combines two key principles: a feature extraction backbone and transfer learning technique.**

Keywords— NeuroCorgi, feature extractor, quantization, transfer learning.

I. Introduction

Low power and low latency AI inference are essential requirements for many applications, such as drone navigation, mechanical weeding and augmented reality. It must be done at the edge for safety, latency, power and privacy reasons. Yet large Neural Network models are difficult to integrate on-chip in an energy-efficient manner. The main issue lies in the large number of parameters, which requires an external memory, leading to a large power dissipation because of the data movement. Numerous approaches exist to mitigate or diminish the impact of these memory accesses. This can be achieved through quantization [1], near calculation weights generation [2, 3], or through computational scheduling that favors the utilization of nearby memory [4]. All these optimization methods are generally applicable to a vast majority of programmable architectures.

But it is possible to push the energy boundary even further, by exploiting the concept of transfer learning and completely revisiting the architecture of accelerators. Transfer learning is a technique in machine learning which allows reusing the knowledge of a pre-trained model and applying it to new tasks. This has gained popularity in Natural Language Processing, where retraining a Foundation model from scratch would be prohibitive. The strategy of reusing feature extraction was introduced in [5], where the authors use the backbone of a classification-trained network, and replace the final classifier layers by a regression network and train it to predict object bounding boxes for localization and object detection task. When new database has limited number of samples, it was proven that this strategy gives better and more stable performance, than training the full network from scratch.

From the point of view of a hardware accelerator, it is therefore possible to freeze the learned general knowledge of a neural network and keep a customizable part adapted for each new task. The first part, the backbone, computes the features of the input data while the latter, the head, addresses the task output (e.g. classification, segmentation, object detection). A new class of accelerator is thus proposed: a Feature Extractor Accelerator (FEA), which is a specialized module to compute the backbone. By fixing the topology and the weights of a FEA, it is possible to minimize both the computing energy and latency while continuing supporting multiple applications.

Fig. 1 depicts this proposal, where a 'fixed weights and fixed topology' FEA is combined with a programmable AI architecture, on an embedded AI computing platform. The FEA is generated at design-time with either a generic database (e.g. ImageNet) or a domain specific database. An implementation flow is used for the training and generation of the FEA part. Thus, it is possible to regenerate a specific FEA accelerator when the generic one does not achieve the required accuracy.

In this paper, a FEA architecture (NeuroCorgi_core) using the MobileNet v1 topology and trained with ImageNet is

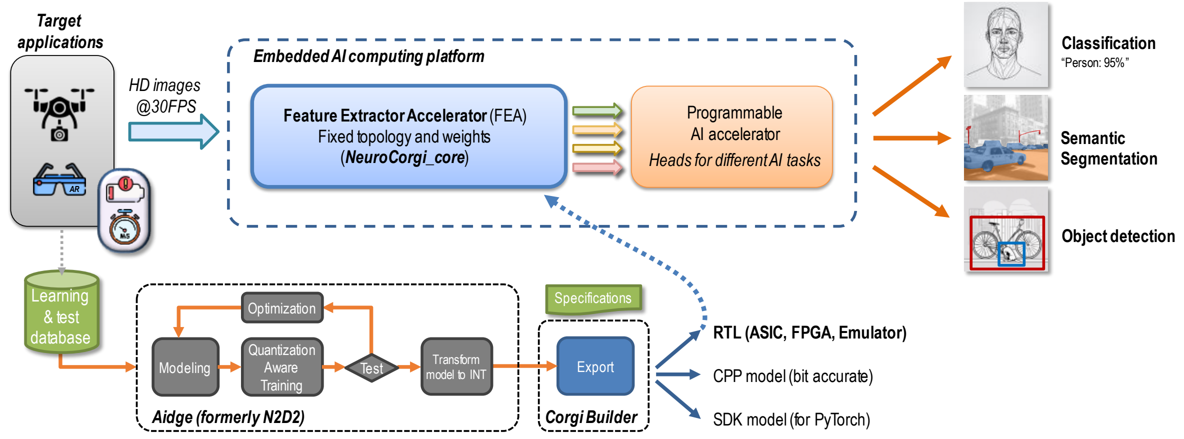


Fig. 1. Embedded AI computing platform for low-power low-latency AI applications. Our proposed fixed-topology fixed-weights feature extractor accelerator (FEA) coupled with a programmable accelerator to address multiple AI tasks. Implementation and training flow for the FEA.

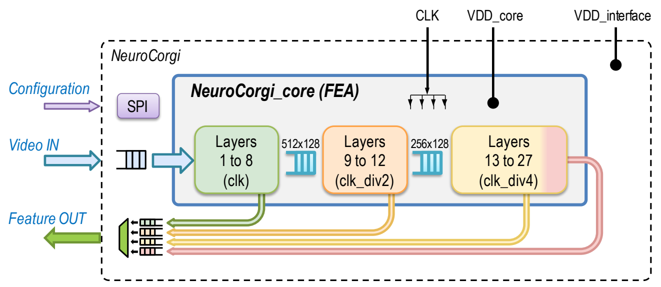


Fig. 2. NeuroCorgi circuit containing the feature extractor accelerator (FEA). No external memory interface, only video IN and feature OUT.

presented, based on early concepts from [6]. It uses a streaming architecture requiring no external memory accesses. The topology and weighs are fixed: 4-bit quantization is applied to weights and activations. On ImageNet, it achieves 70.42% accuracy and sustains native HD image inference at 30FPS with just 23.2mW.

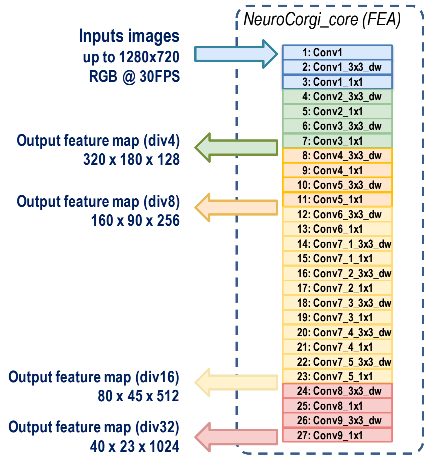
II. NeuroCorgi Overview

NeuroCorgi (Fig. 2) is a FEA demonstrator circuit supporting RGB images of up to 1280x720 pixels. It implements a MobileNet v1 FEA (NeuroCorgi_core) with 27 convolution layers (Fig. 3). Computations occur without external memory usage, whether for weights or activations. The foundational principle revolves around eliminating the energy cost of weights memory accesses by directly freezing them into hard-wired lookup tables (HW-LUT). Consequently, weights are merged with multiplication operators to form Multi-Constant Multipliers (MCMs).

For automating the generation of the NeuroCorgi circuit, a tool called CorgiBuilder was developed: it generates the RTL code, a CPP simulation model and the SDK for PyTorch integration. The generated architecture is customized at design time with multiple parameters including the operator type, the level of pipelining, the parallelization factor, bit precision and saturation operations. To achieve low-latency computing with batch size 1, the architecture uses a layer-wise streaming approach. Moreover, the inter-layer buffers are minimized using a Channel-Width-Height (CWH) traversal approach as no data transposition occurs during the computation (Fig.4).

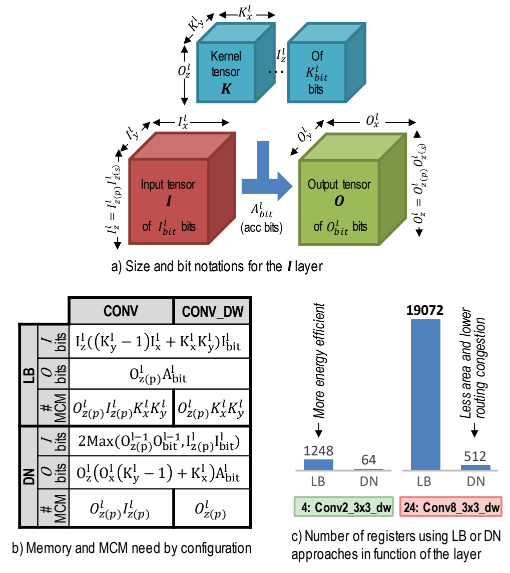
The MCM operators are optimized at design time and at logic synthesis level. Their hardware and energy cost depend on data dynamics and on the table radix [7]. To minimize the data dynamics, the activations and weights are pruned and quantized to 4 bits thanks to the Aidge [8] platform, using the quantization aware training (QAT) method SAT [1]. The resulting model obtains 70.42% classification accuracy on ImageNet. On the other hand, the MCM radix depends on the computing parallelism of the network. The higher the computing parallelism, the lower the MCM radix, but also the higher the area of the computing network. Thus, an optimal point between area, computing parallelism and energy consumption needs to be defined. CorgiBuilder tool allows exploring all these combinations to obtain an optimal implementation.

MobileNet v1 topology uses convolutions (Conv) and depth–wise convolutions (Conv_DW). Both convolution types can be implemented using two modes of data storage: one storing inputs activations in the form of a line buffer (LB) (Fig 5c/d), and the other storing accumulation results (DN variant) (Fig. 5a/b). LB keeps input data to perform a patch traversal of the input tensor (I^l), while DN computes with input pixel through z ($I^l_{Z(p)}$) and stores accumulation results to perform a patch traversal on the output tensor (O^l). Thus, LB stores the input activations in registers and SRAM memories while DN stores accumulations results on SRAM memories (ACC SRAM in Fig 5a/b). For a (K^l_x, K^l_y) convolution layer, both modes store $(K^l_y - 1)$ lines plus (K^l_x) pixels elements of data. Thus, the storage increases linearly with the width of the input image. Moreover, in terms of data width, the input activations used by LB require less bits (e.g. 4bits) rather than accumulation results (e.g. 10bits) of DN. In addition, LB is partially implemented using registers to perform parallel read operations (Fig 5c). Thus, LB performing fewer and smaller read/write operations per MAC, it requires less energy per MAC operation, but this comes at the cost of higher number of registers than DN (Fig. 4b). This number depends on kernel tensor dimension. For deeper layers (>512 features per pixel), LB is unfeasible due to wire-routing congestion (Fig. 4c). Fig 3 shows the CorgiBuilder optimal point with the selected operator mode and the bit precision used on the current implementation.



Weight bits	Activ. bits	Acc. bits	Computational mode
8	4	20	LB Conv
4	4	12	LB Conv DW
4	4	10	LB Conv
4	4	11	LB Conv DW
4	4	12	LB Conv
4	4	10	LB Conv DW
4	4	11	LB Conv
4	4	11	LB Conv DW
4	4	11	LB Conv
4	4	10	LB Conv DW
4	4	10	LB Conv
4	4	10	DN Conv DW
4	4	11	LB Conv
4	4	10	DN Conv DW
4	4	10	LB Conv
4	4	10	DN Conv DW
4	4	10	LB Conv
4	4	10	DN Conv DW
4	4	10	LB Conv
4	4	10	DN Conv DW
4	4	10	LB Conv
4	4	10	DN Conv DW
4	4	10	LB Conv
4	4	10	DN Conv DW
4	4	11	LB Conv
4	4	11	DN Conv DW
4	4	10	LB Conv

Fig. 3. MobileNet v1 (α=1) topology and parameters implemented on NeuroCorgi circuit. Output feature depends on input image size..


K^l_y
K^l_x
O^l_z
Kernel tensor
K
I^l_z
Of K^l_{bit} bits
I^l_y
I^l_x
$I^l_z = I^l_{z(p)} I^l_{z(s)}$
Input tensor
I
of I^l_{bit} bits
A^l_{bit}
(acc bits)
O^l_y
O^l_x
Output tensor
O
of O^l_{bit} bits
$O^l_z = O^l_{z(p)} O^l_{z(s)}$
a) Size and bit notations for the I layer
b) Memory and MCM need by configuration
More energy efficient
Less area and lower routing congestion
1248
64
19072
512
LB
DN
LB
DN
4: Conv2_3x3_dw
24: Conv8_3x3_dw
c) Number of registers using LB or DN approaches in function of the layer

		CONV	CONV_DW
LB	I bits	$I^l_z((K^l_y - 1)I^l_x + K^l_x K^l_y) I^l_{bit}$	
	O bits	$O^l_{z(p)} A^l_{bit}$	
	# MCM	$O^l_{z(p)} I^l_{z(p)} K^l_x K^l_y$	$O^l_{z(p)} K^l_x K^l_y$
DN	I bits	$2\text{Max}(O^{l-1}_{z(p)} O^{l-1}_{bit}, I^l_{z(p)} I^l_{bit})$	
	O bits	$O^l_z(O^l_x(K^l_y - 1) + K^l_x) A^l_{bit}$	
	# MCM	$O^l_{z(p)} I^l_{z(p)}$	$O^l_{z(p)}$

Fig. 4. (a) Layer notation, (b) MCM requirements and (c) register complexity of DN and LB on DW layer.

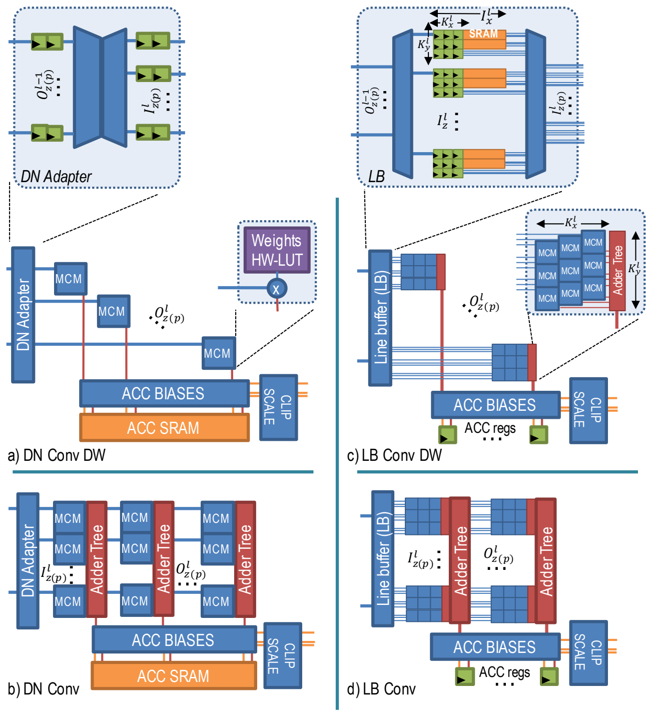


Fig. 5. Compute modes using DN (a)(b) and LB (c)(d) approaches for regular and depth-wise (DW) layers.

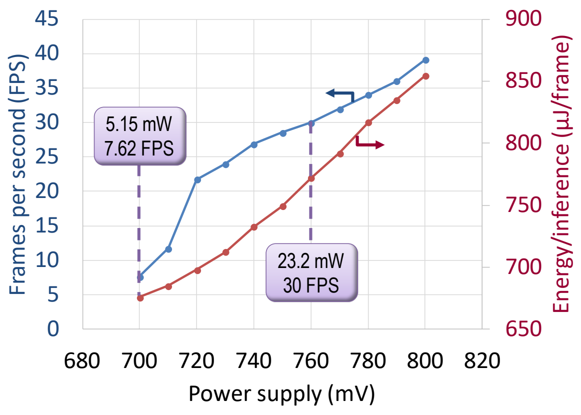


Fig. 7. Measures of energy per inference and FPS on 1280x720 images

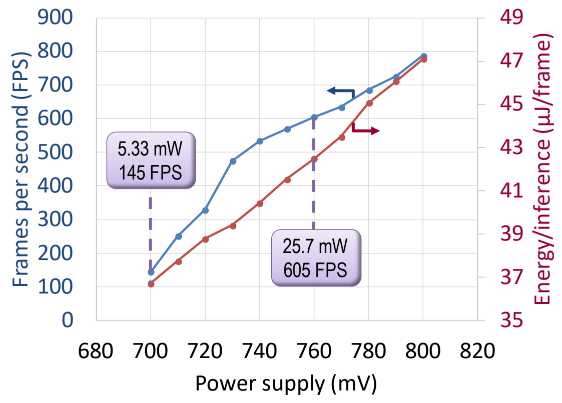


Fig. 8. Measures of energy per inference and FPS on 224x224 images.

SRAM data operations are high energy consuming w.r.t MAC energy operation [14]. Thus, data reuse is key to minimize energy. Computing convolution operations with MCM requires less energy than with conventional multipliers and SRAM memories as the read energy of HW-LUT is much lower than SRAM memory. Therefore, is it possible to minimize the area and compute energy of a convolution kernel by reading activation SRAMs fewer times and weights multiple times.

Network parameters in HW-LUTs benefit from synthesis tool optimizations. Thus, MCM operators and adder-trees are then optimized with fixed weight values. In the NeuroCorgi implementation, a layer computes up-to 128MACs in a single clock cycle per MCM. Thus, in order to minimize glitch power on these long combinational paths, the clocks are deskewed. FIFOs and clock dividers are inserted between layers to improve MAC utilization and energy efficiency by avoiding pipeline bubbles due to line stride (Fig. 2). Moreover, FIFO depths are defined to sustain the required framerate. Finally, QAT is implemented using saturation operators, ReLU clipping and fixed-point scaling (Fig. 6).

Several experiments were carried out to confirm the generalization of the transfer learning technique to a wide spectrum of applications [15]. A Cityscapes [16] segmentation task using 1280x640 images was implemented. It achieves 73.3% mIoU when combining NeuroCorgi with a programmable AI architecture (as illustrated in Fig. 1): 96.7% of the total MAC operations are done in NeuroCorgi, the remaining operations are executed in the programmable AI accelerator. The latter is currently emulated thanks to an FPGA.

This proves that a generic backbone, combined with transfer learning, is applicable to a wide range of domains. That approach was further explored: three different variants of NeuroCorgi, targeting different application domains, were fabricated in GF22FDX technology and successfully tested. This further confirms that the approach is flexible.

III. Measurements Results

Our generic ImageNet variant (Fig. 9) contains 42kMAC operators and 186 SRAM memories. The chip area is 7.86mm² while the FEA area is 4.45mm². At 0.76V and 59MHz, it performs continuous feature extraction of HD images at 30FPS with 23.2mW, 772µJ/frame (Fig. 7). The distribution of power consumption is as follows: 81% is allocated to the combinatorial portion, 12% to the SRAMs, and 7% to the sequential logic. At this operating point, FEA has 224µW leakage and 1064µW idle power.

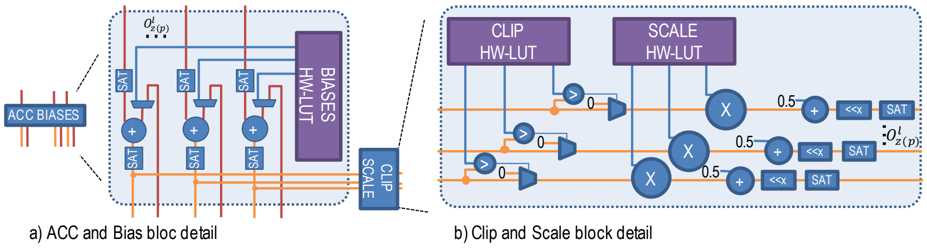
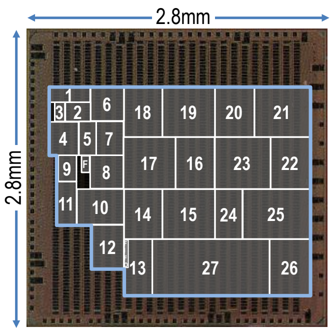


Fig. 6. Detail of (a) accumulation and biases block and (b) clip and scale block

TABLE I. Comparison with state-of-the-art.

			ISSCC'20 [9]	JSSC'23 [10]	JSSC'23 [11] DIANA	JSSC'24 [12] Marsellus	JSSC'24 [13] DynaPlasia	ISSCC'22 [2] Hiddenite	This work NeuroCorgi
Technology			12nm	4nm	22nm	22nm FDX	28nm	40nm	22nm FDX
Application			Server	Mobile	Edge	AI-IoT	Embedded	Embedded	Embedded
Area (mm²)			709 (chip)	4.74 (core)	3.3 (core)	18.7 (chip) 2.42 (core)	20.25 (chip)	9 (chip) 4.36 (core)	7.86 (chip) 4.45 (FEA)
Programmability			Programmable	Programmable	Programmable	Programmable	Programmable	Configurable	Fixed
Power consumption (mW)			25W – 276W	381 – 5133	–	12.8 – 123	261	85.4 – 534.7	5 – 37
ImageNet use case	Training dataset		ImageNet	ImageNet	ImageNet	ImageNet	ImageNet	ImageNet	ImageNet
	AI model		ResNet50 v1	MobileNet TPU	ResNet18	ResNet18	ResNet18	ResNet50	MobileNet v1
	Precision (bits)		8	8	Analog + digital	RBE 4x4b	9w, 8a	ternary(w), 8a	4w, 4a
	Top-1 accuracy (%)		74.93	–	64.1	68.5	70.4	70.09	70.42
	Inferences/second (FPS)	224x224	78563	3433	277	20.8	776^{γ}	169.7^{α}	788
		1280x720	–	–	–	–	–	–	39
	FEA latency (ms)	224x224	$0.2^{\beta\delta}$	0.29^{δ}	3.6^{δ}	48^{δ}	$1.29^{\gamma\delta}$	$5.92^{\alpha\delta}$	1.23
		1280x720	–	–	–	–	–	–	6.90
	TOPS/W	224x224	4.14	11.59	5.52^{α}	5.83	10.8^{γ}	16^{α}	30.9^{ϕ}
		1280x720	–	–	–	–	–	–	30.9^{ϕ}
	Best FEA Energy/inference (µJ/frame)	224x224	2000^{δ}	340^{δ}	$659^{\alpha\delta}$	557^{δ}	$336^{\gamma\delta}$	$503^{\alpha\delta}$	36.7
		1280x720	–	–	–	–	–	–	676

1 MAC = 2 Ops. Zero skipping included as MACs $^{\alpha}$Without considering off-chip memory accesses. $^{\beta}$Latency reported on Inception v3. $^{\phi}$Only feature extraction (no FC layer)
$^{\delta}$Estimated feature extraction part. Assuming 1Conv OP = 1FC OP for latency and energy. FC is 0.18% (0.03%) of total MACs/frame on 224x224 images for MobileNet v1 (ResNet18)
$^{\gamma}$Power & latency off-chip weight loading from external DRAM / external host CPU / on-chip network / on-chip memory access / refresh are not included.



Chip summary	
Technology	GF 22FDX
Chip area	7.86mm²
FEA area	4.45mm²
# multipliers	42k
# SRAM memories	186
SRAM memory	1.1MB
Main clock	59MHz
AI model	MobileNet v1
Training dataset	ImageNet
Batch size	1

Fig. 9. NeuroCorgi die photo with floorplan and chip summary.

When reducing the voltage to 0.7V, the circuit processes 7.62FPS with 5.15mW, 675µJ/frame. At 0.8V, it reaches 39FPS with a power consumption of 33.4mW, 854µJ/frame. The UHVT cells used in the design limit the low voltage operating range of the circuit. On the other hand, the leakage is drastically reduced as it only represents 1% of the total power. Thus, the circuit remains energy efficient at low and high framerates. We also demonstrate feature extraction of 224x224 images at 0.70V and 145FPS, with 5.33mW, 36.7µJ/frame (Fig. 8). At 0.8V and 77MHz, it reaches 788FPS with 37.1mW, 47.1µJ/frame.

The inference energy per input pixel (i.e. energy per inference divided by the input image size) is virtually constant regardless of image size: 733pJ/frame/pixel (HD) and 731pJ/frame/pixel (224x224). The pipelining has been balanced to maximize the computing and energy efficiency regardless of the image size.

The computed latency (i.e. the time between the first input pixel and the first output feature at conv9_1x1) depends on the image size. For HD images at 30FPS, the latency is 6.9ms. This represents 20.7% of the reading time of the image. For 224x224 images at 605FPFS, the latency is 1.28ms (75.8% of image time).

Table I compares this work with prior art. The selected publications report end-to-end energy and latency for ImageNet use case. Thus, the energy and efficiency are reported when computing the full network and not only on the best layer. In the table, the application domains are stated, as the power consumption ranges are extremely different. Server [9] and mobile [10] circuits achieves high framerate with also high power consumptions. Embedded systems [2, 11, 12, 13] reach high-energy efficiency but their reduced internal memory limits this performances to reduced image sizes. The programmability is also reported, as NeuroCorgi has fixed topology and fixed weights. As NeuroCorgi implements only the FEA part (FC layer is not computed), the SoA latency and energy are estimated for the FEA part only (FC on MobileNet v1 on 224x224 images represents 0.18% of the total MAC operations). For a particular architecture, the computing energy efficiency can be increased when the bit precision is reduced at the expense of loss of accuracy. Thus, the work in [13] obtains better energy efficiency at macro level when reducing the bit precision to (5w, 4a). However, neither system-level efficiency nor accuracy is reported for this reduced precision. Finally, none of the compared works reported inference on HD images.

When benchmarking on ImageNet, our fixed architecture achieves at least 1.93× higher energy efficiency (TOPS/W) and 9.2× lower energy per frame (µJ/frame), while also supporting HD image resolution up to 39FPS.

IV. Conclusion

To meet the needs of powerful, yet ultra-low power and low latency AI accelerators at the edge, we combine a fixed Feature Extraction backbone and the concept of transfer learning, for retraining the programmable output heads. This led to the design and fabrication of the NeuroCorgi circuit, which is in itself the first instance of a new class of accelerators: the Feature Extraction Accelerators. It is, to the best of our knowledge, the lowest power-consuming accelerator of the state-of-the-art, capable of handling HD images at 30 FPS for 23.2mW.

Acknowledgment

ANDANTE (ECSEL 876925), PREVAIL (DIGITAL 101083307) EU projects. David Briand, Johannes Christian Thiele and Marc Duranton for their contribution.

References

[1] Qing Jin et al, "Towards Efficient Training for Neural Network Quantization," https://doi.org/10.48550/arXiv.1912.10207.

[2] K. Hirose et al., "Hiddenite: 4K-PE Hidden Network Inference 4D-Tensor Engine Exploiting On-Chip Model Construction Achieving 34.8-to-16.0TOPS/W for CIFAR-100 and ImageNet," 2022 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 2022, pp. 1-3, doi: 10.1109/ISSCC42614.2022.9731668.

[3] V. Ramanujan, M. Wortsman, A. Kembhavi, A. Farhadi and M. Rastegari, "What's Hidden in a Randomly Weighted Neural Network?," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 11890-11899, doi: 10.1109/CVPR42600.2020.01191.

[4] Y. -H. Chen, T. -J. Yang, J. Emer and V. Sze, "Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices," in IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 9, no. 2, pp. 292-308, June 2019, doi: 10.1109/JETCAS.2019.2910232.

[5] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus and Yann LeCun, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks", https://doi.org/10.48550/arXiv.1312.6229.

[6] I. Miro-Panades et al., "Meeting the Latency and Energy Constraints on Timing-critical Edge-AI Systems," book Embedded Artificial Intelligence, River Publishers, 2023.

[7] A. K. Oudjida, A. Liacha, M. Bakiri and N. Chaillet, "Multiple Constant Multiplication Algorithm for High-Speed and Low-Power Design," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 2, pp. 176-180, Feb. 2016, doi: 10.1109/TCSII.2015.2469051.

[8] https://projects.eclipse.org/projects/technology.aidge

[9] Y. Jiao et al., "7.2 A 12nm Programmable Convolution-Efficient Neural-Processing-Unit Chip Achieving 825TOPS," 2020 IEEE International Solid-State Circuits Conference - (ISSCC), San Francisco, CA, USA, 2020, pp. 136-140, doi: 10.1109/ISSCC19947.2020.9062984.

[10] J. -S. Park et al., "A Multi-Mode 8k-MAC HW-Utilization-Aware Neural Processing Unit With a Unified Multi-Precision Datapath in 4-nm Flagship Mobile SoC," in IEEE Journal of Solid-State Circuits, vol. 58, no. 1, pp. 189-202, Jan. 2023, doi: 10.1109/JSSC.2022.3205713.

[11] P. Houshmand et al., "DIANA: An End-to-End Hybrid DIgital and ANAlog Neural Network SoC for the Edge," in IEEE Journal of Solid-State Circuits, vol. 58, no. 1, pp. 203-215, Jan. 2023, doi: 10.1109/JSSC.2022.3214064.

[12] F. Conti et al., "Marsellus: A Heterogeneous RISC-V AI-IoT End-Node SoC With 2–8 b DNN Acceleration and 30%-Boost Adaptive Body Biasing," in IEEE Journal of Solid-State Circuits, vol. 59, no. 1, pp. 128-142, Jan. 2024, doi: 10.1109/JSSC.2023.3318301.

[13] S. Kim et al., "DynaPlasia: An eDRAM In-Memory Computing-Based Reconfigurable Spatial Accelerator With Triple-Mode Cell," in IEEE Journal of Solid-State Circuits, vol. 59, no. 1, pp. 102-115, Jan. 2024, doi: 10.1109/JSSC.2023.3319962.

[14] Bill Dally, "To ExaScale and Beyond," SuperComputing 2010.

[15] I. Miro-Panades et al., "A Multi-Application Platform Supporting Several Uses Cases in the Domains Digital Farming and Transport and Smart Mobility," European Conference on EDGE AI Technologies and Applications (EEAI), Oct. 2023 Athens, Greece.

[16] https://www.cityscapes-dataset.com