To Spike or Not To Spike: A Digital Hardware Perspective on Deep Learning Acceleration

Fabrizio Ottati*† , *Graduate Student Member IEEE*, Chang Gao‡ , *Member IEEE*, Qinyu Chen§ , *Member IEEE*, Giovanni Brignone† , *Graduate Student Member IEEE*, Mario R. Casu† , *Senior Member IEEE*, Jason K. Eshraghian¶ , *Member IEEE* and Luciano Lavagno† , *Senior Member IEEE*

***Abstract*—As deep learning models scale, they become increasingly competitive from domains spanning from computer vision to natural language processing; however, this happens at the expense of efficiency since they require increasingly more memory and computing power. The power efficiency of the biological brain outperforms any large-scale deep learning (DL) model; thus, neuromorphic computing tries to mimic the brain operations, such as spike-based information processing, to improve the efficiency of DL models. Despite the benefits of the brain, such as efficient information transmission, dense neuronal interconnects, and the co-location of computation and memory, the available biological substrate has severely constrained the evolution of biological brains. Electronic hardware does not have the same constraints; therefore, while modeling spiking neural networks (SNNs) might uncover one piece of the puzzle, the design of efficient hardware backends for SNNs needs further investigation, potentially taking inspiration from the available work done on the artificial neural networks (ANNs) side. As such, when is it wise to look at the brain while designing new hardware, and when should it be ignored? To answer this question, we quantitatively compare the digital hardware acceleration techniques and platforms of ANNs and SNNs. As a result, we provide the following insights: (i) ANNs currently process static data more efficiently, (ii) applications targeting data produced by neuromorphic sensors, such as event-based cameras and silicon cochleas, need more investigation since the behavior of these sensors might naturally fit the SNN paradigm, and (iii) hybrid approaches combining SNNs and ANNs might lead to the best solutions and should be investigated further at the hardware level, accounting for both efficiency and loss optimization.**

***Index Terms*—Artificial Neural Networks, Deep Learning, Digital Hardware, Neuromorphic Computing, Spiking Neural Networks.**

I. Introduction

From smartphones to televisions and cars, deep learning (DL) has become pervasive in our daily lives. Many modern DL models, especially large language models (LLMs) [1], recommender systems [2], and vision transformers [3] require huge amounts of power and energy for both training and inference. For instance, Vit-G/14 [4], a top performing model in object recognition, requires 2.86 GFLOP for a single inference on ImageNet [5]. The model contains 184.3 billions of parameters, and optimizing these parameters has also a non negligible environmental impact [6]. In fact, Vit-G/14 has a training time of $3 \cdot 10^4$ TPUv3 days [4]; given that a TPUv3 consumes an average of 220 W[7], the energy consumption for training can be estimated to be 159 MW h. This training is performed on 220 W highly-efficient and specialized TPUs, while the brain runs within a 20 W power budget and it is able to perform multiple tasks at once.

Tackling these challenges requires more efficient neural network models and hardware. Many neuromorphic engineers are looking at how the brain might offer us a blueprint for making DL more efficient [8]. This is because the brain is capable of causal reasoning, integrating various sensory modalities in executing long-term planning, and providing sensorimotor feedback to enable us to engage with our environments actively. Moreover, it does so far more efficiently than any combination of large-scale DL models currently available. The pursuit of brain-inspired computing has triggered a surge in the popularity of spiking neural networks (SNNs) [9] with the ultimate goal of realizing efficient artificial intelligence (AI). For instance, SpikeGPT [10], the first spiking LLM, is estimated to use $22\times$ fewer operations in its execution when compared to its conventional non-spiking counterpart. Thus, model optimizations may ultimately outweigh the low-level hardware issues discussed in this paper.

At the inception of neuromorphic computing, analog-domain computation was shown to be a promising substrate for the deployment of brain-inspired hardware. The metal-oxide-semiconductor transistor in the sub-threshold regime can emulate the diffusion-based dynamics of neuronal ion channels. At the same time, memristive technologies reproduce key features of biological neurons [11], [12]. However, analog designs suffer from scalability issues due to transistor mismatch and noise, and long design time due to reduced electronic design automation (EDA) tool support [13]. In contrast, in deep sub-micron technologies, digital designs benefit from strong EDA support and reliable operation. While research on analog computation and in-memory processing has flourished over the past decade, digital accelerators are easier to design and deploy in the immediate short term.

This paper quantitatively reviews the broad landscape of digital accelerators for both SNN and conventional artificial neural network (ANN) accelerators. To analyze spike-based processing pipelines in the machine learning context, we segment the analysis into static and temporal (e.g., sequence learning) workloads as follows:

* Corresponding author: fabrizio.ottati@polito.it
† Department of Electronics and Telecommunications Engineering, Politecnico di Torino, Torino, Italy.
‡ Department of Microelectronics, Delft University of Technology, Delft, Netherlands.
§ Institute of Neuroinformatics, University of Zürich and ETH Zürich, Zürich, Switzerland.
¶ University of California Santa Cruz, Santa Cruz, California, USA.

- Section II analyzes the low-level neuron models employed in SNNs and ANNs.
- Section III introduces the metrics and used to compare the hardware accelerators from the ANN and SNN domains, and an energetic model to approximately estimate how static and sequential tasks would perform on SNN and ANN hardware.
- Section IV analyzes and compares the hardware acceleration of convolutions for static vision workloads (in particular, object recognition), and then compares state-of-the-art (SOTA) digital ANN and SNN accelerators.
- Section V analyzes and compares the hardware acceleration of temporal workloads, i.e., tasks that involve sequences of inputs evolving in time.

We derive the following conclusions:

- Based on the current SOTA digital chips results and measurements on static data classification tasks, ANNs perform better than SNNs in processing efficiency and classification accuracy. Since ANNs Pareto dominate SNNs, we claim that SNNs do not suit static tasks. A key reason is that SNN classification requires multiple timesteps, resulting in more operations than ANN since the latter is computed in a single pass.
- On temporal tasks, SNNs show energy efficiency comparable to ANN chips; furthermore, only in the SNN domain there is an example of on-chip training and learning on temporal sequences with competitive task performance [14]; hence, further investigation in this direction is needed, together with model training techniques to improve performance (e.g., classification accuracy) and system energy consumption.
- Classification-based workloads employing bio-inspired sensors data, such as event cameras [15] and silicon cochleas [16], naturally fit the stateful nature of spiking neurons, but SOTA models and accelerators are not exploring this kind of tasks, which could lead to an effective advantage related to SNNs models.
- The optimal solution for a given classification task might be a heterogeneous model, made of ANN and SNN parts that operate synergically together. Further research is needed in this direction.

II. Neuron Models

A basic introduction to the SNN and ANN models employed in this analysis is provided. Using these models, a quantitative estimation of the energy consumption of these neurons on vision and sequence learning tasks is made.

A. The spiking neuron

The discrete leaky integrate and fire (LIF) neuron model is among the most commonly used models in the literature [9], and is described by:

$$v_{l,i}[t] = \beta \cdot v_{l,i}[t-1] + u_{l,i}[t] - \vartheta \cdot S_{l,i}[t-1] \tag{1}$$

$v_i[t]$ is the neuron state, β is the decay coefficient associated with the leakage, and $S_i[t]$ is the output spike. The l suffix denotes the l-th layer in the network. The input current $u_{l,i}[t]$ is given by:

$$u_{l,i}[t] \triangleq \sum_j w_{l,ij} \cdot S_{l-1,j}[t] \tag{2}$$

The output spike is modeled by:

$$S_i[t] = \begin{cases} 1 & \text{if } v_i[t] \geq \vartheta \\ 0 & \text{otherwise} \end{cases}$$

$S_i[t]$ is equal to 1 at spike time (i.e., if at timestamp t the state $v_i[t]$ is larger than the threshold ϑ) and 0 elsewhere.

Note that since $S_{l-1,j}[t]$ is either 0 or 1, the input current $u_{l,i}[t]$ is the sum of the synaptic weights of the $l-1$-th layer neurons spiking at timestamp t.

Equation (1) requires three inputs to update the state $v[t]$: the weights w_{ij}, the previous value $v[t-1]$ and the input spikes $S_j[t]$. This means that an SNN hardware processing element (PE) needs access to 3 memory structures to retrieve the inputs, the state, and the weights.

B. The artificial neuron

The most common artificial neuron model [17] is static in time, i.e., it does not preserve any state between successive inputs. A particular class of ANNs, namely recurrent neural networks (RNNs), employ special gates that introduce a state in the neuron [17]. Here, the stateless ANN neuron is analyzed while in Section V, recurrent cells are discussed.

The artificial neuron model is:

$$z_{l,i} = \varphi(\sum_j z_{l-1,j} \cdot w_{l,ij} + b_{l,i}) \tag{3}$$

The weights are multiplied by the inputs from the previous layer, $z_{l-1,j}$, and accumulated; an optional bias term $b_{l,j}$ is added in (3). An activation function, $\varphi(x)$, is applied to the accumulated value. The specific choice of activation function often depends on the application, layer, and the type of neural network [17].

Distinct from (1), only two inputs are needed in (3): the previous layer activations $z_{l-1,j}$ and the current layer weights $w_{l,ij}$. This implies that only two memory structures are needed for the artificial PE being implemented in hardware: this represents an advantage over SNN implementations in both memory (and area) occupation of the circuit, and energy consumption, since memory accesses are the most energy-intensive operations in modern digital hardware architectures[18].

III. Methodology

In ANN architectures, the most common operation is the multiply and accumulate (MAC) [19], which includes the input activation multiplication by the corresponding weight and the subsequent accumulation. We count each MAC as two operations, since it computes a multiplication and an addition, even if in hardware they might be merged. This is to isolate the contribution of each computation to total power consumption, and is the same approach adopted by the ANN hardware research considered in this analysis [20]–[23]. In

TABLE I: Energy consumption comparison between integer add, multiply and memory operation on on-chip static random access memory (SRAM) caches [18], targeting a 45 nm CMOS process.

	Energy [pJ]	**Energy density** [pJ/B]
Add 8 b	0.03	0.03
Multiply 8 b	0.20	0.20
Read 64 b (8 kB capacity)	10.00	2.50

SNN hardware, no multiplication is performed [9]: a weight is accumulated if there is an input spike, otherwise it is not. Hence, this is considered a single operation in our analysis.

In the SNN literature, there are numerous references to the synaptic operation (SynOP) metric [24], [25]. However, there is little consensus on its formal definition. Given the ambiguous definition of this metric, it is not used to measure the efficiency of an SNN accelerator in this work. In addition to ambiguity, the SynOP metric does not tend to account for stateful operations (state access and updates).

To obtain an estimation of the energy cost of SNN and ANN architectures, two simple mathematical models are provided in Sections IV and V. These allow us to approximately compare ANN and SNN hardware accelerators *a priori*. While evaluating the actual hardware performance, considering efficiencies in terms of operations per second per watt (OPS/W) is not enough: on static data, an SNN would need multiple time steps to perform a classification, while an ANN accelerator performs an inference in a single forward-pass. To ensure fairness, this paper adopts the energy consumption per inference as the most balanced metric for comparison; latency and accuracy-related metrics are accounted for separately.

Sparsity is ignored in the energy consumption analysis performed in Sections IV and V, since modern ANN accelerators have complex and very efficient sparse dataflows [20], [26], [27]; however, the sparsity handling capability of SOTA accelerators is included in the energy consumption results shown in Tables II and III.

IV. Static tasks

Equation (2) embeds one of the major advantages of SNN processing, namely the removal of multiplication between the input feature map (i.e., spikes) and the synaptic weights. In theory, this leads to both hardware and energy savings. However, considering the data in Table I, the energy consumption for reading a byte from the on-chip buffer, implemented as SRAM, exceeds the cost of an 8 b integer multiplication. Hence, a multiply-free SNN digital hardware accelerator is not necessarily more efficient than an ANN accelerator, given that SNNs require additional memory accesses to update the neuron states.

A. Energy model

To evaluate these mathematical models quantitatively on static tasks, the convolution operations shown in Algorithms 1 and 2 are used as benchmarks, since convolution is the fundamental operation employed in the large majority of current hardware accelerators for object recognition, detection and so on [22], [23], [28], [29]. In more recent approaches, transformer architectures are emerging as better-performing vision models [30]; as such, a transformer accelerator [20] is considered in the final results section.

Algorithm 1 SNN convolution of a single window and timestamp.

```
Require: $S, \beta, \vartheta$                          ▷ Stride, leakage, threshold.
Require: weights                                         ▷ Convolution kernel weights.
Require: states                                          ▷ Neurons states.
Require: ifmap, ofmap                                    ▷ Input and output feature maps.
Require: $(c_o,\ h_o,\ w_o)$                             ▷ Output value coordinates.
 1: $I \leftarrow 0$                                     ▷ Input synaptic current.
 2: for $c_i \leftarrow 0,\ C_I - 1$ do                  ▷ Input channels.
 3:     for $h_k \leftarrow 0,\ H_K - 1$ do              ▷ Kernel height.
 4:         for $w_k \leftarrow 0,\ W_K - 1$ do          ▷ Kernel width.
 5:             $h_i \leftarrow h_o * S + h_k$
 6:             $w_i \leftarrow w_o * S + w_k$
 7:             if $ifmap[c_i][h_i][w_i] \neq 0$ then
 8:                 $I \leftarrow I + weights[c_o][c_i][h_k][w_k]$
 9:             end if
10:         end for
11:     end for
12: end for
13: $m \leftarrow states[c_o][h_o][w_o] * \beta + I$     ▷ State update.
14: if $m \geq \vartheta$ then
15:     $m \leftarrow m - \vartheta$
16:     $ofmap[c_o][h_o][w_o] = 1$
17: else
18:     $ofmap[c_o][h_o][w_o] = 0$
19: end if
20: $states[c_o][h_o][w_o] \leftarrow m$
```

With respect to Algorithms 1 and 2, all activations, weights, and states are quantized to 8 b, while spikes are treated as unary quantities. Consider Algorithm 1:

- $C_I \cdot H_K \cdot W_K$ weights and spikes are read from memory, where C_I is the number of channels in the input feature map, and H_K and W_K represent the shape of the convolution kernel. For the sake of clarity, this quantity is denoted with N_{rd}.

$$N_{rd} \triangleq C_I \cdot H_K \cdot W_K$$

  Assuming that 8 spikes are encoded to an 8 b memory word in the spike scratchpad (i.e., the memory structure used to host the input and output data near the PE), the energy associated with a spike memory operation (either read or write) can be approximated to $E_{rd}/8$, where E_{rd} (E_{wr}) is the energy consumption of a memory read (write) considering the 8 kB cache field in Table I. The energy consumption associated with the memory accesses of weights and spikes is denoted with $E_{rd_{tot}}$:

$$E_{rd_{tot}} = N_{rd} \cdot (E_{rd} + E_{rd}/8)$$

- $C_I \cdot H_I \cdot W_I$ additions are performed on the synaptic current. The energy associated to checking if the spike

($ifmap[c_i][h_i][w_i]$) is equal to 1 is negligible since its cost is included in the memory access performed to retrieve the spikes. Hence, this energy is denoted with E_{acc}.

$$E_{acc} = N_{rd} \cdot E_{add}$$

- Next, one state has to be retrieved from memory, multiplied by the leakage, β, added to the synaptic current, thresholded (compared against the threshold, ϑ, and, if higher, reduced by ϑ via subtraction) and written back. In the worst case, this energy denoted with E_{state}, is given by:

$$E_{state} = E_{rd} + E_{mult} + E_{add} + E_{comp} + E_{sub} + E_{wr}$$

- The output spike must then be written to the scratchpad. This energy is denoted with E_{ofmap}.

$$E_{ofmap} = E_{wr}/8$$

Hence, the total energy involved in an SNN convolution, defined as E_{SNN}, is:

$$E_{SNN} = E_{rd_{tot}} + E_{acc} + E_{state} + E_{ofmap}$$

The following values are employed for numerical estimation: the shape of the convolution, i.e., of the input feature map window to be evaluated in order to compute a single output feature map value, is $(C_I, H_I, W_I) = (512, 3, 3)$, which means the number of memory reads is $N_{rd} = 4608$. For memory operations, additions/subtractions, and multiplications, the data from Table I are used. It has to be remarked that, in this analysis, we do not consider any memory optimization (e.g., buffering on registers) since both ANNs and SNNs access the memories with the same window patterns; thus, these would bring similar advantages to both of them, without impacting the comparison. The energy consumed by a threshold comparison, E_{comp}, is assumed to be the same as that of an 8 b addition. This is a reasonable assumption, since comparison commonly employs a subtraction. This leads to:

$$E_{SNN} = 13.1\,\text{nJ}$$

The energy consumption obtained above assumes a dense input feature map, i.e., all input neurons are firing. To take into account sparse firing activity, a coefficient γ_{SNN} can be introduced that reflects the proportion of neurons in the feature map that are firing at a given time, and the hardware overhead due to the sparse data structures employed. Hence:

$$E_{SNN} = 13.1\,\text{nJ} \cdot \gamma_{SNN}$$
$$0 < \gamma_{SNN} \leq 1$$

Consider now the convolution operation performed in an ANN, which is reported in Algorithm 2.

Following the same approach adopted for the SNN convolution:

- N_{rd} weights and inputs are read from memory:

$$E_{rd_{tot}} = 2N_{rd} \cdot E_{rd}$$

Algorithm 2 ANN convolution of a single window.

```
1: $a \leftarrow 0$                                          ▷ Activation.
2: for $c_i \leftarrow 0,\ C_I - 1$ do
3:     for $h_k \leftarrow 0,\ H_K - 1$ do
4:         for $w_k \leftarrow 0,\ W_K - 1$ do
5:             $h_i \leftarrow h_o * S + h_k$
6:             $w_i \leftarrow w_o * S + w_k$
7:             $a \leftarrow a +$
8:                 $weights[c_o][c_i][h_k][w_k] * ifmap[c_i][h_i][w_i]$
9:         end for
10:    end for
11: end for
12: $z = \varphi(a)$                                  ▷ Non-linear activation.
13: $ofmap[c_o][h_o][w_o] = \psi(z)$                          ▷ Quantisation.
```

- The same number of additions and multiplications are performed, and this energy is denoted with E_{MAC}.

$$E_{MAC} = N_{rd} \cdot (E_{add} + E_{mult})$$

- The obtained value is then processed by the nonlinear activation function $\varphi(z)$. This energy is denoted with E_{act}.
- The result is quantized in order to be processed by the next layer in the network. The quantization step is modeled through a function $\psi(z)$, with an associated energy cost E_{quant}, which is estimated as an 8 b addition (for instance, the rectified linear unit (ReLU) activation is a 0-thresholding).
- Finally, the result is written to the scratchpad memory:

$$E_{ofmap} = E_{wr}$$

The total energy consumption of an ANN convolution is:

$$E_{ANN} = 24.1\,\text{nJ} \cdot \gamma_{ANN}$$

A similar sparsity coefficient can be introduced, denoted with γ_{ANN}. The approximations made for E_{quant} and E_{act} are acceptable since their contribution to the total energy is negligible, given that they are performed only on the final result of the convolution.

Hence, despite the additional memory accesses and state operations involved in SNNs, ANN convolutions consume $1.84\times$ more energy. Of course, this result depends heavily on the convolution filter depth and size, but it gives a reasonable approximation of the different costs between ANN and SNN processing, as highlighted in Section IV-B.

However, the energy estimation obtained for the SNN corresponds to a single time step! When dealing with static data, such as images, an SNN needs multiple time steps (T, for instance), since the input image pixels are encoded to multiple spikes in time [9]. Hence, the actual energy consumption associated with a convolution operation in an SNN must be multiplied by the number of time steps needed to perform an inference:

$$E_{SNN} = 13.1\,\text{nJ} \cdot T \cdot \gamma_{SNN}$$

Since $T > 1$ for any SNN, otherwise there would be no temporal evolution in the network and it would be equal to

TABLE II: DL accelerators evaluated on ImageNet [5]. The best efficiency is considered for each design, and the associated task accuracy is evaluated under the same conditions. Mixed indicates an accelerator running both SNN and ANN processing elements.

	ANN Transf.	**ANN CNN**		**SNN**	**Mixed**
Work	Keller'23 [20]	Park'22 [23]	Mo'21 [21]	SNPU'23 [28]	C-DNN'23 [29]
Process [nm]	5	7	28	28	28
Area [mm^2]	0.2	4.7	1.9	6.3	20.3
Supply voltage [V]	1.1	1.0	0.9	1.1	1.1
Clock frequency [MHz]	1760	1196	470	200	200
Data format	INT4-VSQ [20]	INT8	INT8	INT8, INT4	INT1-16, INT4/8
Network model	DeiT-Base	MobileNetTPU	ResNet50	ResNet18	ResNet50
Parameters [M]	768	3	25	12	25
Operations/inference [GOP]	35.2	17.4	13.3	61.4	7.4
Task accuracy [%]	80.5	71.7	76.9	66.8	77.1
Throughput [FPS]	56	3433	120	245	123
Power [mW]	56	5114	132	478	34
Energy/inference [mJ]	1.0	1.5	1.1	2.0	0.3

heavily quantized ANNs, SNNs are less efficient on static vision data than ANNs, if the same degree of sparsity-awareness ($\gamma_{SNN} = \gamma_{ANN}$) is taken into account. This conclusion is validated by the data shown in Table II, which includes SOTA accelerators at the time of writing. The given results account for how the hardware handles sparse feature maps, as stated by the author of the papers considered [20]–[23], [28], [29].

B. SOTA accelerators

The literature provides many overviews of hardware accelerators for SNNs, ranging from digital hardware to in-memory computing (IMC) and mixed-signal architectures [24], [31], [32]. An up-to-date list of SNN hardware accelerators and processors can be found in [33]. In these reviews [24], different coding schemes for SNNs, such as latency coding and phase coding [9], are explored; however, these still lack in classification accuracy performance with respect to rate coding, even if they represent an interesting approach that could provide an advantage to SNNs.

Most SNN hardware reviews are missing an important feature: an objective comparison with SOTA ANN hardware accelerators. Most SNN accelerators are benchmarked on static datasets; such datasets are inappropriate for proper system characterization, since (i) they are often considered trivial or "solved" datasets (e.g., CIFAR-10, MNIST) [24], and (ii) ANNs are more efficient on static data.

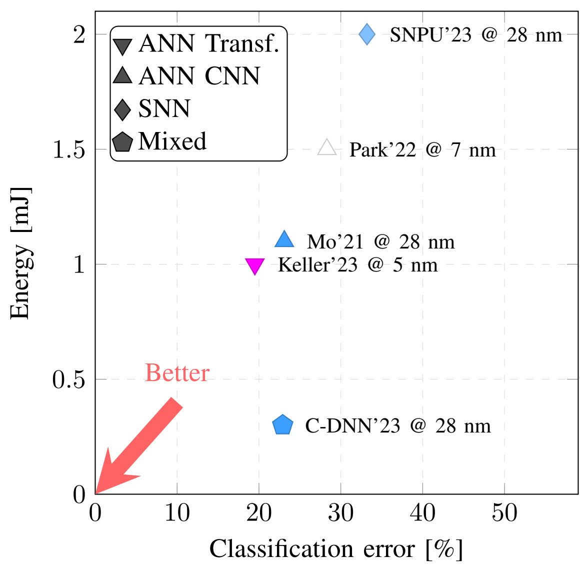
Table II reports SOTA ANN vision accelerators and the best performing SNN accelerator [28] (SNPU'23). All accelerators run the same task: object classification on ImageNet [5]. SNPU'23 [28] is chosen as the SNN reference since it is the only accelerator targeting complex vision workloads such as ImageNet. In addition to SNPU'23, C-DNN'23 [29] is analyzed. This chip employs both ANN and SNN PEs to maximize inference efficiency by inferring part of the neural network layers on the SNN hardware and part of these on the ANN hardware. In fact, mixed neural models employing both artificial and spiking backbones are arising as high performance implementations which allow to improve SNNs accuracy [34], [35] on vision tasks, beyond tackling more complex workloads such as object detection. Moreover, C-DNN'23 also provides on-chip training capabilities, which most ANN accelerators lack.

Table II considers the most efficient SNN digital hardware accelerator, SNPU'23 [28], and a mixed-topology design, C-DNN'23 [29]. These are compared to various ANN accelerators for object recognition, which target both transformer-based models [30] and convolutional neural networks [36]. Different observations can be made:

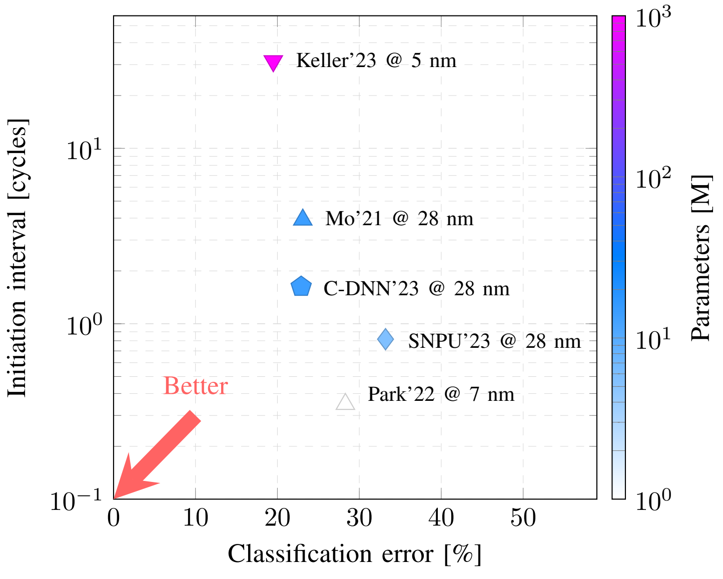
- The model employed by SNPU'23 [28], ResNet18 [36] SNN, is the lowest performing model in terms of classification accuracy: $66.8\,\%$, against $80.5\,\%$ (Keller'23 [20]), $71.68\,\%$ (Park'22 [23]), $76.92\,\%$ (Mo'21 [21]) and $77.1\,\%$ (C-DNN'23 [22]).
- The energy per inference of SNPU'23 [28] is the highest among all the designs ($1.95\,\mathrm{mJ/inf}$), even considering an ANN accelerator implemented on the same $28\,\mathrm{nm}$ complementary metal-oxide-semiconductor (CMOS) node, i.e., Mo'21 [21] ($0.46\,\mathrm{mJ/inf}$).

Figure 1 shows the energy consumption per inference and the performance vs. the top-1% classification error on ImageNet, for each accelerator reported in Table II. The performance is expressed as the initiation interval, i.e., the ratio between the clock frequency and the throughput. It is the number of clock cycles between two inferences at the output at the steady state and it corresponds to the latency of non-pipelined accelerators. It is worth noting that this performance metric is totally architecture-dependent and allows for more immediate comparison between accelerators implemented with different technological process. While SNPU'23 is considered the SOTA SNN accelerator at the time of writing, it is nonetheless Pareto-dominated by all ANN accelerators, both in the energy vs. error (Fig. 1a) and in the performance vs. error (Fig. 1b) spaces. This is because SNPU'23 requires 16 timesteps to process a single input image; hence, even if the SNPU'23 model contains less than half of the parameters of the Mo'21 one, it requires $4.6\,\times$ operations per inference.

For what concerns C-DNN'23 [29], the mixed architecture leads to a very low energy consumption per inference ($281\,\mu\mathrm{J}$), the best among all the chips despite being implemented on a CMOS node older than Keller'23 [20] and Park'22 [23]



(a) Energy.



(b) Performance.

Fig. 1: Energy and performance of DL accelerators with respect to classification error on ImageNet.

($28\,\mathrm{nm}$ vs. $5\,\mathrm{nm}$ and $7\,\mathrm{nm}$, respectively). However, the model being run on C-DNN'23, the ResNet50, is much smaller than the vision transformer of Keller'23, which achieves a higher accuracy (80.5% vs. 77.1%).

The design presented in SNPU'23 is among the best-performant in the SNN domain. Accelerating ResNet family models on-chip is an impressive feat, given that many SNN accelerators are limited to smaller-scale architectures; however, our analysis of SNPU'23 against ANN accelerators highlights that static data is not the optimal way to demonstrate processing efficiency. Beyond static vision tasks, a possibility is to focus on tasks that take advantage of the event-based nature of SNNs, such as dynamic vision sensor (DVS) data [15]. These sensors capture scenes in the form of 'events' that can be treated naturally as spikes, without any artificial encoding. Some alternative SNN accelerators target dynamic workloads, though only those that handle large-scale (at least on the scale of ResNets) models with static data are considered in the above analysis [14], [37]. These accelerators represent the minority [24], since most benchmarks are limited to the MNIST and CIFAR-10 static datasets.

Beyond classification, more complex event-based vision tasks are less explored by the neuromorphic community, though tend to dominate the modern computer vision ANN field [38]–[41]. Conversely, on-chip learning solutions in the SNN domain [14], [25] are more advanced than for the ANN community. This advantage should be exploited to reduce training costs and allow for adaptive intelligence at the edge. In order to reduce the memory access burden for computation of neuron states, low-rate training techniques should be explored for efficient hardware inference [9], [25].

It has to be remarked that on-chip training solutions have been and are investigated in the ANN domain [42]; however, these approaches target general purpose platforms, such as microcontrollers, and still target simple neural networks and tasks. Of course, this is true also for SNNs: in fact, the classification accuracy of on-chip trained networks is worse than the one of off-chip models, which perform worse than the ANN counterpart. This is why in practically any application, inference-only, ANN-based models are deployed on highly efficient accelerators.

V. Temporal tasks

SNNs are inherently time-aware neural networks due to their statefulness (see Section II). As such, they are a natural fit for sequential data processing. In video processing tasks, such as video segmentation, both non-spiking and spiking neural networks often employ convolution structures to extract features [43]–[45]. Given that the computational costs of spiking and non-spiking convolutional operators are addressed in Section IV, this section primarily concentrates on audio processing, another prominent subset of temporal tasks.

In previous work [46], [47], a fully connected feed-forward RNN targeting keyword prediction is used to compare ANNs, rate-based SNNs, and latency-based SNNs [9]. The rate-based SNN is only $9\,\%$ more efficient than the ANN due to the high input firing rate ($2.5\,\mathrm{kHz}$) necessary to match the ANN performance. The limited efficiency advantage makes the effort of migrating to a new type of neural network hard to justify. Conversely, the latency-based SNN is $84\,\%$ more efficient than the ANN, primarily thanks to the significantly lower firing rate that leads to a reduction in the number of the timesteps evaluated by the SNN. However, this methodology sets the target time window to $75\,\mathrm{ms}$ to incorporate temporal information, which is not suitable for real-time processing.

To evaluate the performance and efficiency of ANN and SNN accelerators, in the following audio processing benchmarks are considered, such as keyword spotting (KWS), voice activity detection (VAD) and automatic speech recognition (ASR) [48]–[50]. In particular, we present an energy analysis similar to the one in Section IV. Finally, we compare SOTA digital hardware accelerators from the spiking and artificial domains.

A. RNN versus SNN

RNNs are designed for discerning patterns in sequential data, uniquely characterized by their capacity to retain memory of previous inputs within their hidden state. Variants that aim to enhance the "memory" of such neurons have also emerged, such as lost short-term memories (LSTMs) and gated recurrent units (GRUs) [17], and have been adopted in audio processing tasks [51], [52]. The formulation of a vanilla RNN layer is:

$$\begin{aligned} h_t &= \sigma_h(U_h \cdot x_t + V_h \cdot h_{t\text{-}i} + b_h) \\ o_t &= \sigma_o(W_o \cdot h_t + b_o) \end{aligned} \tag{4}$$

where x is the input, h the hidden layer, and o the output. U_h, W_o, and V_h are the weight matrices, b is the bias vector and σ is the activation function. Differently from the SNN model defined by (1), the next state is computed considering a bias b_h and by multiplying the previous state by a matrix V_h; in SNNs, V_h reduces to a scalar β, employed for all the neurons [9]. The equivalent synaptic current is represented by $U_h \cdot x_t$.

We compare the computational cost of these operations in the context of a speech recognition task. We consider the cost of a vanilla RNN and an SNN, which share the same mechanism of implicit recurrence through the neuron state [13], [17]. As in Section IV, activations, weights, and states are quantized to 8 bits, while spikes are single-bit quantities. As a use case, we consider a layer with N inputs and calculate the energy consumption of a neuron in the layer, as in Section IV.

With the SNN model:

1) N weights and input spikes are loaded from memory:

$$E_{rd_{tot}} = N \cdot (E_{rd} + E_{rd}/8)$$

2) These values are then accumulated depending on the spike value to obtain the activation to be fed to the state:

$$E_{acc} = N \cdot E_{add}$$

3) The state is loaded from memory and decayed (i.e., multiplied) by a factor β; then, it is accumulated with the activation computed in the previous step, compared against the threshold ϑ, reset if needed and stored to memory:

$$E_{state} = E_{rd} + E_{mult} + E_{add} + E_{comp} + E_{sub} + E_{wr}$$

4) The output spike, if generated, is then stored to the scratchpad:

$$E_{ofmap} = E_{wr}/8$$

Assuming $N = 1024$ inputs and considering the data reported in Table I, the total energy, E_{SNN}, is given by:

$$E_{SNN} = 2.92\,\text{nJ} \cdot \gamma_{SNN}$$

Consider now the vanilla artificial RNN layer:

1) N weights and N inputs are read from memory, together with the hidden state and its recurrent weight:

$$E_{rd_{tot}} = (2N + 2) \cdot E_{rd}$$

2) the inputs and the state are multiplied by the weights and accumulated:

$$E_{MAC} = (N + 1) \cdot (E_{add} + E_{mult})$$

3) the state is written back to memory. As in Section IV, the quantization and activation energies are neglected:

$$E_{state} = E_{wr}$$

4) the obtained value is processed by the nonlinear activation function, quantized and written back to memory:

$$E_{ofmap} = E_{wr}$$

Hence, the total energy consumption is:

$$E_{ANN} = 5.37\,\text{nJ} \cdot \gamma_{ANN}$$

The energy analysis shows that in both vanilla RNN and spiking layers, the memory accesses for weights and states consume the majority of the energy, as in the convolution case. Notice that E_{SNN} is $1.84\times$ smaller than E_{ANN}: this is due to the fact that while in the ANN the inputs are on $8\,\text{b}$, in the SNN these are single-bit quantities. Since the memory access energy dominates the other figures, this results in a major overhead of ANN models with respect to SNN ones. Moreover, the ANN model involves a multiplication when processing inputs, which is more energy-hungry than the addition (Table I).

Differently from the convolution case, the number of timesteps needed to process the inputs is larger than 1 for both ANNs and SNN, since the data is now evolving through time.

It has to be taken into account that very simple RNN and SNN models are considered. In Section V-B, digital hardware accelerator targeting more sophisticated neural network architectures are investigated.

B. SOTA accelerators

Regarding audio processing tasks, most accelerators are tested on or designed specifically for VAD [53], [54] and KWS [14], [55]–[57]. This is due to the fact that VAD and KWS represent less challenging problems to tackle; hence, simple neural network architectures are employed, which allow achieving higher efficiency on hardware inference.

In Table III we chose the two most efficient SNN chips [14], [53], respectively in VAD and KWS, which are compared against SOTA ANN counterparts. For the VAD task, Oh'19 [54] achieves the best efficiency measured in energy per inference, despite being implemented on an old $180\,\text{nm}$ CMOS technology node. It results to be more efficient and to perform better, in terms of classification accuracy, than Yang'19 [53], which runs an SNN model and uses the same CMOS node.

As for the KWS task, different CMOS technologies are employed in the accelerators. These values are not normalized with Dennard scaling since all the chips have a power consumption in the order of $10\,\mu\text{W}$; in these conditions, most

TABLE III: SNN and ANN accelerators evaluated on temporal tasks including VAD and KWS.

	VAD		KWS			
	SNN	ANN	SNN	ANN		
Work	Yang'19 [53]	Oh'19 [54]	Frenkel'22 [14]	Kim'22 [55]	Giraldo'22 [56]	Shan'23 [57]
Process [nm]	180	180	28	65	65	28
Area [mm^2]	2.57	17.55	0.45	2.03	2.56	3.6
Supply voltage [V]	0.55	0.6	0.5	0.75	0.6	0.4
Clock frequency [MHz]	0.5	0.7	13	0.25	0.25	0.2
Feature extractor	Analog	Analog	No FEx	Analog	Digital	Digital
Data format	INT1	INT4	INT8	INT8	INT8	INT1
Dataset	Aurora4 w/ DEMAND	LibriSpeech w/ NOISEX-92	Spiking Heidelberg Digits	GSCD		
Task accuracy [%]	85	90	90.7 (1-word)	86 (10-word)	90.9 (10-word)	97.8 (2-word)
Network model	FCN	FCN	Spiking RNN	GRU	LSTM	DSCNN
Parameters [k]	4.6	1.6	132	24	21.5	4.7
Power [μW]	1	0.142	79	23	10.6	0.8
Energy/inference [nJ]	10	2.3	42	285.2	169.6	23.6
Latency [ms]	-	512.0	5.7	12.4	16.0	29.5

of the power consumption is static, while Dennard scaling gives an approximation of how dynamic power scales across technology nodes.

Also for KWS, the highest energy efficiency is achieved by an ANN chip, Shan'23 [57], which also has a lower average power consumption and significantly higher accuracy than all the other chips targeting the same task. The SNN accelerator, Frenkel'22 [14], achieves the lowest latency and is the only chip that supports online learning and on-chip training: in fact, the whole training process is performed on-chip, that is validated also on vision and autonomous agent tasks. It should be noted that the number of parameters of the spiking RNN in [14] is $28\times$ larger than that of the ANN chip in [57], while achieving lower accuracy. This might suggest that SNNs are still lacking in terms of classification accuracy when compared to their ANN counterparts; however, Shan'23 [57] is an inference only chip, hence the model is fine-tuned for the task, while Frenkel'22 supports arbitrary network topologies and on-chip training and can be repurposed. This is a significant advantage for chips deployed on edge devices, which model might need to be re-adapted to the environment in which the system is deployed.

Figure 2 shows the distribution of these accelerators in terms of energy efficiency and accuracy along with FPGA accelerators [27], [58], that achieve the highest accuracy in KWS. One can notice that Shan'23 [57] Pareto-dominates all the other chips, thanks to a highly efficient sparsity-aware chip architecture and a performant neural network model. Frenkel'22 [14] presents a very competitive efficiency, taking into account that it is the only chip with in-hardware training capabilities: in fact, the network benchmarked on KWS is trained directly on it; nonetheless, the resulting classification accuracy on the task results to be competitive even when considering most of the ANN chips analyzed.

VI. Conclusions

This paper analyzes SOTA digital hardware accelerators implementing ANN and SNN models performing two types of tasks: static and temporal datasets. The architectures are compared on classification accuracy and energy consumption per inference. The results suggest that:

- Current ANN models and digital hardware accelerators outperform their SNN counterparts on static images and object recognition tasks. One reason is that the multi-timestep processing of SNNs increases the operations per inference, leading to throughput and energy overheads.
- On temporal tasks, SNNs show a really competitive energy efficiency, and represent the only case in which full on-chip training and learning is performed [14]. This advantage should be exploited, together with new training strategies that promote both improvement in classification accuracy, which is still lower than the ANN counterpart, and sparse firing activity, that would lower further the energy consumption of the accelerator.
- Further investigation of efficient model and hardware solutions targeting bio-inspired sensors data, such as event

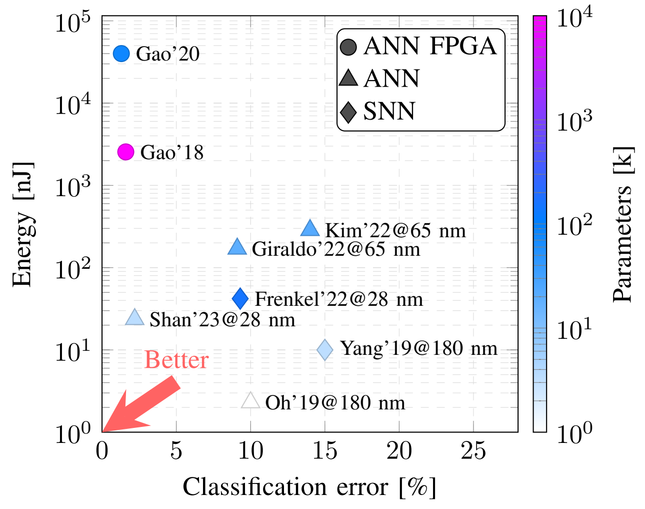


Fig. 2: Energy vs. classification error of recent digital SNN and ANN accelerators measured on the VAD and KWS tasks.

cameras [15] and silicon cochleas [16], are needed to take advantage of the time-related functioning of spiking neurons and architectures.
- Hybrid SNN and ANN solutions might be the key to maximize task performance and efficiency. C-DNN'23 [29] shows the second-best efficiency among the accelerators analyzed, despite being implemented in a $28\,\mathrm{nm}$ CMOS process, while the best-performing chip takes advantage of a $5\,\mathrm{nm}$ CMOS technological node.

In conclusion, the answer to our original question is that there are few tasks in which one should spike, but full on-chip learning accelerators, such as ReckOn [14], and mixed architectures represent a promising research direction that should be further addressed, instead of simply replicating ANN architectures on tasks where there is no efficiency or accuracy advantage to SNNs.

Acknowledgments

This work was partially supported by the Key Digital Technologies Joint Undertaking under the REBECCA Project with grant agreement number 101097224, receiving support from the European Union, Greece, Germany, Netherlands, Spain, Italy, Sweden, Turkey, Lithuania, and Switzerland.

The authors would like to thank: Kim Sangyeob, who is with the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, for providing the energy per inference measurements for C-DNN'23 [29].

References

[1] T. Brown *et al.*, "Language models are few-shot learners," in *Adv. Neural Inf. Process. Syst.*, 2020, pp. 1877–1901.

[2] M. Naumov *et al.*, *"Deep learning recommendation model for personalization and recommendation systems"*, 2019. arXiv: 1906.00091.

[3] A. Dosovitskiy *et al.*, *"An image is worth 16x16 words: Transformers for image recognition at scale"*, 2020. arXiv: 2010.11929.

[4] X. Zhai *et al.*, "Scaling vision transformers," in *Conf. Comput. Vis. Pattern Recog.*, 2022, pp. 12 104–12 113.

[5] J. Deng *et al.*, "Imagenet: A large-scale hierarchical image database," in *Conf. Comput. Vis. Pattern Recog.*, 2009, pp. 248–255.

[6] A. Fu *et al.*, "Reconsidering CO2 emissions from Computer Vision," in *Proc. IEEE/CVF Conf. CVPR*, 2021, pp. 2311–2317.

[7] Google Cloud, *"System Architecture of TPU VM"*, https://cloud.google.com/tpu/docs/system-architecture-tpu-vm?hl=en#tpu_v3, (accessed Jun. 27, 2023).

[8] S. Schmidgall *et al.*, *"Brain-inspired learning in artificial neural networks: a review"*, 2023. arXiv: 2305.11252.

[9] J. K. Eshraghian *et al.*, *"Training spiking neural networks using lessons from deep learning"*, 2021. arXiv: 2109.12894.

[10] R.-J. Zhu *et al.*, *"Spikegpt: Generative pre-trained language model with spiking neural networks"*, 2023. arXiv: 2302.13939.

[11] R. Douglas *et al.*, "Neuromorphic analogue VLSI," *Annu. Rev. Neurosci.*, vol. 18, no. 1, pp. 255–281, Feb. 1995.

[12] M. Payvand *et al.*, *"Dendritic Computation through Exploiting Resistive Memory as both Delays and Weights"*, 2023. arXiv: 2305.06941.

[13] J. K. Eshraghian *et al.*, "Memristor-based binarized spiking neural networks: Challenges and applications," *Nanotechnol. Mag.*, vol. 16, no. 2, pp. 14–23, Apr. 2022.

[14] C. Frenkel and G. Indiveri, "ReckOn: A 28nm sub-mm2 task-agnostic spiking recurrent neural network processor enabling on-chip learning over second-long timescales," in *Int. Solid-State Circuits Conf.*, 2022, pp. 1–3.

[15] G. Gallego *et al.*, "Event-based vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 154–180, Jan. 2020.

[16] M. Yang *et al.*, "A 0.5 V 55μW 64x2 Channel Binaural Silicon Cochlea for Event-Driven Stereo-Audio Sensing," *IEEE J. Solid-State Circuits*, vol. 51, no. 11, pp. 2554–2569, Nov. 2016.

[17] I. Goodfellow *et al.*, *Deep learning*. MIT press, 2016.

[18] H. Mark, "Computing's energy problem (and what we can do about it)," in *Int. Solid-State Circuits Conf.*, 2014, pp. 9–13.

[19] V. Sze *et al.*, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[20] B. Keller *et al.*, "A 95.6-TOPS/W Deep Learning Inference Accelerator With Per-Vector Scaled 4-bit Quantization in 5 nm," *IEEE J. Solid-State Circuits*, vol. 58, no. 4, pp. 1129–1141, Apr. 2023.

[21] H. Mo *et al.*, "9.2 A 28nm 12.1 TOPS/W dual-mode CNN processor using effective-weight-based convolution and error-compensation-based prediction," in *Int. Solid-State Circuits Conf.*, 2021, pp. 146–148.

[22] Y. Wang *et al.*, "A 28nm 27.5 TOPS/W approximate-computing-based transformer processor with asymptotic sparsity speculating and out-of-order computing," in *Int. Solid-State Circuits Conf.*, 2022, pp. 1–3.

[23] 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," *IEEE J. Solid-State Circuits*, vol. 58, no. 1, pp. 189–202, Jan. 2023.

[24] A. Basu *et al.*, "Spiking neural network integrated circuits: A review of trends and future directions," in *Custom Integr. Circuits Conf.*, 2022, pp. 1–8.

[25] C. Frenkel *et al.*, "Bottom-Up and Top-Down Approaches for the Design of Neuromorphic Processing Systems: Tradeoffs and Synergies Between Natural and Artificial Intelligence," *Proc. IEEE*, vol. 111, no. 6, pp. 623–652, Jun. 2023.

[26] C. Gao *et al.*, "Spartus: A 9.4 TOp/s FPGA-Based LSTM Accelerator Exploiting Spatio-Temporal Sparsity," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–15, Jun. 2022.

[27] C. Gao *et al.*, "EdgeDRNN: Recurrent Neural Network Accelerator for Edge Inference," *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 10, no. 4, pp. 419–432, Dec. 2020.

[28] S. Kim *et al.*, "SNPU: An Energy-Efficient Spike Domain Deep-Neural-Network Processor With Two-Step Spike Encoding and Shift-and-Accumulation Unit," *IEEE J. of Solid-State Circuits*, vol. PP, pp. 1–14, Jan. 2023.

[29] S. Kim *et al.*, "C-DNN: A 24.5-85.8 TOPS/W complementary-deep-neural-network processor with heterogeneous CNN/SNN core architecture and forward-gradient-based sparsity generation," in *Int. Solid-State Circuits Conf.*, 2023, pp. 334–336.

[30] S. Khan *et al.*, "Transformers in vision: A survey," *ACM comput. surv.*, vol. 54, no. 10s, pp. 1–41, Sep. 2022.

[31] M. Bouvier *et al.*, "Spiking neural networks hardware implementations and challenges: A survey," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 15, no. 2, pp. 1–35, Apr. 2019.

[32] M. R. Azghadi *et al.*, "Hardware implementation of deep network accelerators towards healthcare and biomedical applications," *IEEE Trans. Biomed. Circuits Syst.*, vol. 14, no. 6, pp. 1138–1159, Dec. 2020.

[33] F. Ottati, *Awesome Neuromorphic Hardware*, https://github.com/fabrizio-ottati/awesome-neuromorphic-hw, 2023.

[34] X. She *et al.*, "SAFE-DNN: A Deep Neural Network With Spike Assisted Feature Extraction For Noise Robust Inference," in *Int. Jt. Conf. Neural Netw.*, 2020, pp. 1–8.

[35] B. Chakraborty *et al.*, "A fully spiking hybrid neural network for energy-efficient object detection," *IEEE Trans. Image Process.*, vol. 30, pp. 9014–9029, Oct. 2021.

[36] K. He *et al.*, "Deep residual learning for image recognition," in *Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 770–778.

[37] A. Di Mauro *et al.*, "SNE: an energy-proportional digital accelerator for sparse event-based convolutions," in *Design Autom. Test Eur. Conf. Exhib.*, 2022, pp. 825–830.

[38] E. Perot *et al.*, "Learning to detect objects with a 1 megapixel event camera," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 16 639–16 652, Dec. 2020.

[39] L. Cordone *et al.*, "Object detection with spiking neural networks on automotive event data," in *Int. Jt. Conf. Neural Netw.*, 2022, pp. 1–8.

[40] M. Gehrig and D. Scaramuzza, "Recurrent vision transformers for object detection with event cameras," in *Conf. Comput. Vis. Pattern Recog.*, 2023, pp. 13 884–13 893.

[41] S. Barchid *et al.*, *"Spiking neural networks for frame-based and event-based single object localization"*, 2022. arXiv: 2206.06506.

[42] J. Lin *et al.*, *"On-device training under 256kb memory"*, 2022. arXiv: 2206.15472.

[43] K. Xu *et al.*, "Spatiotemporal CNN for video object segmentation," in *Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 1379–1388.

[44] Q. Chen *et al.*, "Reducing latency in a converted spiking video segmentation network," in *Int. Symp. Circuits Syst.*, 2021, pp. 1–5.

[45] Q. Chen *et al.*, "Skydiver: A Spiking Neural Network Accelerator Exploiting Spatio-Temporal Workload Balance," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 41, no. 12, pp. 5732–5736, Dec. 2022.

[46] G. Chen *et al.*, "Small-footprint keyword spotting using deep neural networks," in *Int. Conf. Acoust. Speech Signal Process.*, 2014, pp. 4087–4091.

[47] B. U. Pedroni *et al.*, "Small-footprint Spiking Neural Networks for Power-efficient Keyword Spotting," in *Biomed. Circuits Syst. Conf.*, 2018, pp. 1–4.

[48] B. Cramer *et al.*, "The heidelberg spiking data sets for the systematic evaluation of spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 7, pp. 2744–2757, Jul. 2020.

[49] P. Warden, *"Speech commands: A dataset for limited-vocabulary speech recognition"*, 2018. arXiv: 1804.03209.

[50] V. Panayotov *et al.*, "Librispeech: an asr corpus based on public domain audio books," in *Int. Conf. Acoust. Speech Signal Process.*, 2015, pp. 5206–5210.

[51] C. Gao *et al.*, "Real-Time Speech Recognition for IoT Purpose using a Delta Recurrent Neural Network Accelerator," in *Int. Symp. Circuits Syst.*, 2019, pp. 1–5.

[52] J. Li *et al.*, "High-accuracy and low-latency speech recognition with two-head contextual layer trajectory LSTM model," in *Int. Conf. Acoust. Speech Signal Process.*, 2020, pp. 7699–7703.

[53] M. Yang *et al.*, "Design of an Always-On Deep Neural Network-Based 1- μ W Voice Activity Detector Aided With a Customized Software Model for Analog Feature Extraction," *IEEE J. Solid-State Circuits*, vol. 54, no. 6, pp. 1764–1777, Jun. 2019.

[54] S. Oh *et al.*, "An Acoustic Signal Processing Chip With 142-nW Voice Activity Detection Using Mixer-Based Sequential Frequency Scanning and Neural Network Classification," *IEEE J. of Solid-State Circuits*, vol. 54, no. 11, pp. 3005–3016, Sep. 2019.

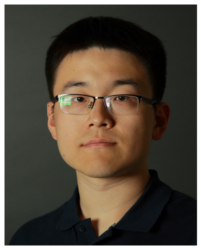
[55] K. Kim *et al.*, "A 23-μW Keyword Spotting IC With Ring-Oscillator-Based Time-Domain Feature Extraction," *IEEE J. Solid-State Circuits*, vol. 57, no. 11, pp. 3298–3311, Nov. 2022.

[56] J. S. P. Giraldo *et al.*, "Vocell: A 65-nm Speech-Triggered Wake-Up SoC for 10- μ W Keyword Spotting and Speaker Verification," *IEEE J. Solid-State Circuits*, vol. 55, no. 4, pp. 868–878, Apr. 2020.

[57] W. Shan *et al.*, "AAD-KWS: A Sub-μ W Keyword Spotting Chip With an Acoustic Activity Detector Embedded in MFCC and a Tunable Detection Window in 28-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 58, no. 3, pp. 867–876, Mar. 2023.

[58] C. Gao *et al.*, "DeltaRNN: A Power-Efficient Recurrent Neural Network Accelerator," in *Int. Symp. Field-Prog. Gate Arrays*, 2018, pp. 21–30.

Fabrizio Ottati (Graduate Student Member IEEE), received the M.Sc. in Electronic Engineering from Politecnico di Torino in 2020. He is currently pursuing the Ph.D. degree in Electrical, Electronics and Telecommunication Engineering in the same university, under the supervision of Professor Luciano Lavagno. His research interests are digital hardware design and machine learning.



Chang Gao (Member IEEE), received his Ph.D. degree with Distinction in Neuroscience from the Institute of Neuroinformatics, University of Zürich and ETH Zürich, Zürich, Switzerland, in March 2022. In August 2022, he joined the Delft University of Technology, The Netherlands as an Assistant Professor in the Department of Microelectronics. He received the 2022 Misha Mahowald Early Career Award in Neuromorphic Engineering, the 2022 Marie-Curie Postdoctoral Fellowship and the title of 2023 MIT Technology Review Innovators Under 35 in Europe. His current research interest is in digital edge machine learning hardware accelerator design and its applications in next-gen telecommunications, video/audio processing, healthcare and robots.

Qinyu Chen (Member IEEE), received the B.S. degree in Communication Engineering from Shandong University, Jinan, China in 2016, and the Ph.D. degree in Microelectronics from Nanjing University, Nanjing, China, in 2021. She is now a postdoctoral researcher at the Institute of Neuroinformatics, University of Zürich and ETH Zürich, Zurich, Switzerland, and an incoming Assistant Professor with the Leiden University, Leiden, The Netherlands. Her current research interest includes the seamless neuromorphic artificial intelligence system at the edge, and its application in healthcare, AR/VR with a focus on event-based processing. In 2022, She received a Bridge Fellowship Grant from the Swiss National Science Foundation (SNSF) and Innosuisse.

Giovanni Brignone (Graduate Student Member, IEEE) received the M.Sc. degree in computer engineering from the Politecnico di Torino, Italy, in 2021, where he is currently pursuing the Ph.D. degree with the Department of Electronics and Telecommunications under the supervision of Professor Luciano Lavagno. His research interests include high-level synthesis, digital hardware design, and HW/SW co-design.

Mario R. Casu (Senior Member, IEEE) received the Ph.D. degree in electronics and communications engineering from Politecnico di Torino, Torino, Italy, in 2001. He is currently an Associate Professor with Politecnico di Torino. His research interests include systems-on-chip (SoC) with specialized accelerators, system-level design and design methodology for FPGAs and ASICs, and embedded machine learning. He is also interested in the design of circuits, systems, and platforms for industrial applications, such as biomedical, automotive, and food. His past work focused on the latency-insensitive design of SoC and networks-on-chip. He regularly serves on the Technical Program Committee for international conferences, such as DAC, ICCAD, and DATE.

Jason K. Eshraghian (Member IEEE) received the B.Eng. (electrical and electronic), L.L.B., and Ph.D. degrees from The University of Western Australia, Perth, WA, Australia, in 2016 and 2019, respectively. From 2019 to 2022, he was a Post-Doctoral Research Fellow at the University of Michigan, Ann Arbor MI, USA. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, The University of California at Santa Cruz, Santa Cruz, CA, USA. His research interests include neuromorphic computing, resistive random access memory (RRAM) circuits, and spiking neural networks.

Luciano Lavagno (Senior Member, IEEE) received the Ph.D. degree in electrical engineering and computer science from UC Berkeley, in 1992. He was an Architect with POLIS HW/SW Co-Design Tool. From 2003 to 2014, he was an Architect with Cadence CtoSilicon High-Level Synthesis Tool. Since 1993, he has been a Professor with Politecnico di Torino, Italy. He has coauthored four books and over 200 scientific papers. His research interests include the synthesis of asynchronous circuits, HW/SW co-design, high-level synthesis, and design tools for wireless sensor networks.