Are SNNs really more energy-efficient than ANNs? An in-depth hardware-aware study

Manon Dampfhoffer, Thomas Mesquida, Alexandre Valentian, Lorena Anghel

Are SNNs Really More Energy-Efficient Than ANNs? An In-Depth Hardware-Aware Study

Manon Dampfhoffer*†, Thomas Mesquida†, Alexandre Valentian†, Lorena Anghel*
*Univ. Grenoble Alpes, CEA, CNRS, Grenoble INP, INAC-Spintec, 38000 Grenoble, France
†Univ. Grenoble Alpes, CEA, List, F-38000 Grenoble, France
Email: manon.dampfhoffer@cea.fr

***Abstract*—Spiking Neural Networks (SNNs) hold the promise of lower energy consumption in embedded hardware due to their spike-based computations compared to traditional Artificial Neural Networks (ANNs). The relative energy efficiency of this emerging technology compared to traditional digital hardware has not been fully explored. Many studies do not consider memory accesses, which account for an important fraction of the energy consumption, use naïve ANN hardware implementations, or lack generality. In this paper, we compare the relative energy efficiency of classical digital implementations of ANNs with novel event-based SNN implementations based on variants of the Integrate and Fire (IF) model. We provide a theoretical upper bound on the relative energy efficiency of ANNs, by computing the maximum possible benefit from ANN data reuse and sparsity. We also use the Eyeriss ANN accelerator as a case study. We show that the simpler IF model is more energy-efficient than the Leaky IF and temporal continuous synapse models. Moreover, SNNs with the IF model can compete with efficient ANN implementations when there is a very high spike sparsity, i.e. between 0.15 and 1.38 spikes per synapse per inference, depending on the ANN implementation. Our analysis shows that hybrid ANN-SNN architectures, leveraging a SNN event-based approach in layers with high sparsity and ANN parallel processing for the others, are a promising new path for further energy savings.**

***Index Terms*—Spiking neural networks, neuromorphic hardware, deep neural network accelerators, energy efficiency.**

I. Introduction

Artificial Neural Networks (ANNs) can solve difficult tasks and Deep Neural Networks (DNNs) are now widely used in a variety of applications such as image recognition [1], speech recognition [2] or medical diagnosis [3]. However their hardware implementation in general-purpose processors (CPUs or GPUs) is inefficient in terms of energy consumption. Therefore, researchers are focusing on designing energy-efficient specialized hardware with spatial architectures. An entirely different approach is to use bio-inspired Spiking Neural Networks (SNNs) instead of ANNs. SNNs closely mimick the communication and computation in the brain. They communicate information via temporal events (spikes), and the computation in a neuron consists in accumulating the spikes, weighted by synaptic connections, into a membrane potential and firing an output spike when the potential reaches a threshold. Therefore, they require only accumulate (AC) operations. They have temporal dynamics, meaning the inference phase is performed over a certain period of time in which neurons can fire several spikes, this latency depending on the required network accuracy. ANNs, on the other hand, perform multiply-and-accumulate (MAC) operations between input activations (*iacts*) and weights on a layer-by-layer basis. SNN communication and event-based computation (i.e. only when there is an input spike) can be leveraged in dedicated neuromorphic hardware, such as the large-scale architectures Truenorth [4], Loihi [5], DYNAPs [6] and FPGA implementations [7], [8].

While SNN event-based implementations naturally benefit from spike sparsity, ANN implementations present other advantages. Indeed, an efficient dataflow (i.e. scheduling of ANN computations and mapping across Processing Elements (PEs) [9]), can optimize the data reuse (locally reusing data in several MACs) such that memory accesses are minimized, reducing the energy consumption. SNN event-based implementations can not leverage data reuse due to the non-flexible and non-predictable order of computations driven by spikes. Moreover, ANNs can also leverage the sparsity of *iacts* via data compression and logic to skip unnecessary MAC operations [10].

Brain-inspired SNN implementations hold the promise of massive power savings. However, our analysis shows these energy savings are contingent on a high level of spike sparsity of the SNN. Most algorithmic papers, such as in [11]–[13], take into account only the number of MAC and AC operations to estimate the energy efficiency. Therefore, SNN algorithms are often considered more energy-efficient than their ANN counterparts due to the replacement of MACs by lighter AC operations. However, most of the energy consumption of neural networks in specialized architectures does not come from the arithmetic operations but from the associated memory accesses [14]. Recent studies compare ANNs and SNNs implementation in specialized accelerators [15]–[17]. However, ref. [15], [17] take the MNIST task as reference to compare both implementations, which is not representative of the characteristics such as sparsity, data reuse opportunity, accuracy and network topologies of more difficult tasks. Moreover, ref. [17] bases its comparison on a single ANN accelerator, and the conclusions may not be valid for other accelerators. Ref. [16] compares memory accesses and computations in ANNs and SNNs to determine the sparsity level required in a SNN to be more energy-efficient than an ANN. However, neither ref. [16] nor ref. [15] consider the opportunities to exploit sparsity and data reuse in ANNs. Moreover, ref. [15]–[17] do not consider the different variants of the IF model frequently used in state-of-the-art SNN algorithms, although

the SNN model impacts the energy consumption. Therefore, we currently lack a fair comparison between ANN and SNN hardware implementations.

In this paper, we propose a high fidelity model for comparing the energy efficiency of ANNs and SNNs on digital hardware, considering how data reuse and sparsity play a role. Guidelines for improving the energy efficiency of SNNs are also provided. The main contributions of this paper are summarized as follows:

- We propose a model of ANN and SNN dynamic energy consumption based on the synaptic operations, considering different variants of the IF model for SNNs, which we use to evaluate the energy efficiency of state-of-the-art SNN algorithms (in Section II).
- We provide an upper bound on ANN energy efficiency relative to the SNNs, independent of the hardware architecture, by computing the maximum benefit of data reuse and exploitation of sparsity (in Section III-A).
- We compare this theoretical bound with the state-of-the art ANN accelerators Eyeriss v1 and v2 [10], [18] (in Section III-B).
- We discuss the effectiveness of hybrid ANN-SNN implementations (in Section IV).

II. Models for the energy consumption of ANN and SNN synapses

A. Scope of the study

In this study, we focus on image recognition applications as these are frequently used for benchmarking DNNs. We consider on-chip inference (and not learning). We take into account only convolutional and fully connected layers, which represents the main energy consumption of ANNs and SNNs. Activation functions, typically Rectified Linear Unit (ReLU) for ANNs and comparison with a constant threshold for SNNs, pooling and normalization layers consume relatively little energy.

A neural network hardware architecture can be spatially expanded (each neuron is physically represented) or spatially folded (time-multiplexed) [15], as depicted in Fig. 1. In a spatially expanded architecture, all the memory is on-chip in buffers (such as SRAMs) close to the PEs. In a spatially-folded architecture, which is typically used in ANNs (for example in Eyeriss [18]), the chip is smaller, with only one buffer on-chip, and there is an off-chip memory (such as DRAM). This type of architecture saves area but increases the energy consumption due to the off-chip memory accesses. In the case of SNNs, the choice of hardware architectures depends on the processing mode, whether it is an event-based execution, where all layers are computed in parallel, or ANN-like execution, where layers are computed one at a time. In event-based execution (such as [4]–[6]), spikes are encoded in a packet containing the address of the source neuron (whose size depends on the network topology) that is sent to the destination neuron in real time using the Address Event Representation (AER) protocol [19]. Event-based processing allows leveraging the spike sparsity of SNNs, as spikes are not stored and processed immediately, but requires a spatially expanded architecture. In ANN-like execution, spikes must be stored, but it offers more opportunities for data reuse and allows spatially folded architectures as in ANNs. For instance, ref. [20] proposes an output stationary dataflow where neurons are mapped to PEs to minimize the energy consumption of reading and writing the neurons state. This comes at the cost of storing the spikes in a FIFO during the computation of a layer, which must be further sorted to respect the computation order constraints. In [21], [22], input spikes are stored as ANN activations with only 0 and 1 values, but with an additional temporal dimension (representing SNN temporal dynamics). Therefore, they can process all spikes of a layer in parallel and use a weight stationary or output stationary dataflow. However, they must also store spikes in tensors whose size depends on the temporal resolution.

These approaches provide scalability, but it is unclear whether they provide an energy benefit. Indeed, they lose the advantages of SNNs, i.e. not storing activations and naturally leveraging their sparsity, and still cannot exploit as much data reuse as in an ANN due to the computation order constraints. Therefore, in this study, we focus on the event-based approach. Instead of taking an existing neuromorphic accelerator, we will consider a more general event-based architecture, and therefore spatially expanded, which is not specific to a SNN model and focus only on inference. In spatially expanded architectures, the supported network topologies are limited by the size of the chip. In case of a large network topology, only a fraction of the topology could be processed in a SNN event-based mode. This is not a problem as we assert that SNN event-based processing is only beneficial for the portion of the network with high spike sparsity, as will be seen later. Therefore, to fairly compare ANNs and SNNs, we consider that both use a spatially expanded architecture in order to use the same memory energy values.

We study the dynamic energy consumption associated with the synaptic operations of ANNs and SNNs (computation and memory accesses), which is evaluated considering the sparsity and data reuse. The static energy consumption and energy consumption associated with communication are other factors impacting the overall energy consumption of a system. However, dynamic energy consumption associated with synaptic operations being one of the main motivation for the use of SNNs (due to the assumed benefits of spike sparsity and replacement of MACs by ACs), we decided to focus our study on this point. Thus, static energy consumption and energy consumption associated with communication are out of the scope of this paper. For the same reason, we focus on energy consumption rather than area, latency and throughput, although they are also important factors to consider when designing an accelerator.

This study focuses on digital accelerators, although SNNs could also benefit from low-power analog implementations. In particular, neural networks accelerators based on processing-in-memory are a promising alternative to reduce data movement [23]. However, device variability and non-ideality of analog circuits impose constraints on the network architecture and can reduce the accuracy, and thus are not the focus of this paper. Moreover, although we focused on convolutional

and fully connected topologies for image processing, the conclusions of this study also apply to recurrent topologies. Indeed, while this adds some constraints for the SNN event-based implementation (output spikes of the recurrent layer must be buffered before being fed to the same layer at the next timestep), they do not affect the dynamic energy consumption of synaptic operations considered in this study.

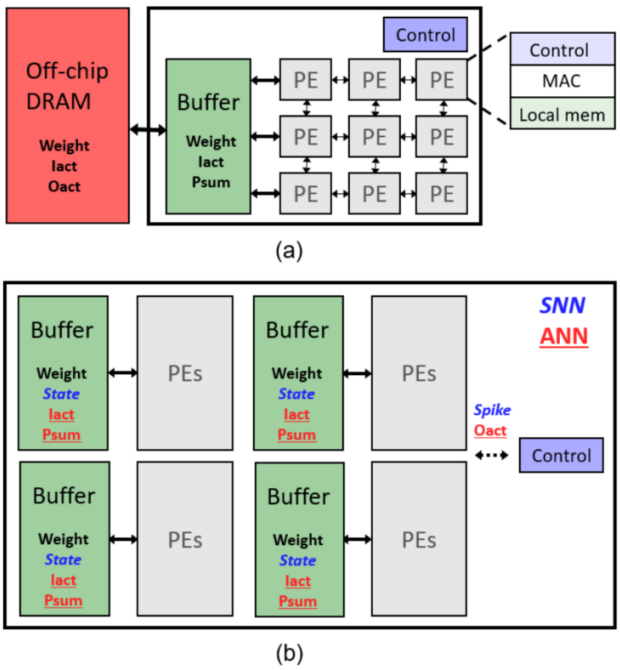

Fig. 1. (a) Spatially folded architecture (such as the Eyeriss v1 chip [18]). (b) Spatially expanded architecture. Data types specific to ANN and SNN are highlighted and in italic, respectively.

B. ANN model

In ANNs, the output of a neuron is defined as:

$$y_i = \varphi(\sum_j x_j w_{ij} + b_i) \tag{1}$$

y_i is the output activation (*oact*) of neuron i, b_i its bias, x_j is the *iact* from presynaptic neuron j, and w_{ij} is the synaptic weight between neurons i and j. φ is an activation function, such as ReLU. Therefore, the ANN atomic operation is the synaptic operation, which corresponds to a MAC operation. Note that the number of synapses (N_{syn}) is different from the number of weights (especially in convolutional architectures where weights are reused in multiple synapses). We start by considering a naïve ANN implementation : for each MAC, we must read the *iact*, weight and current partial sum (*psum*), and write back the updated *psum* [9]. Therefore, ER_x (respectively EW_x) being the energy of the read (respectively write) operation on the x data, the total energy for the ANN is:

$$E_{ANN} = N_{syn} \times (ER_{iact} + ER_{weight} + ER_{psum} + EW_{psum} + E_{MAC}) \tag{2}$$

C. SNN models

For SNNs, we evaluate frequently used variants of the Integrate and Fire model [24]. Neurons can be simple Integrate and Fire (IF), or with an additional leak (LIF), meaning that the membrane potential decays over time. Synapses can be instantaneous or continuous, whether the spike is integrated immediately to the membrane potential or is accumulated first in another state variable which also has a dynamic behavior. Note that only the IF neuron and instantaneous synapse do not have temporal dynamics, while the other models require a temporal discretization to perform the associated updates at each timestep. The different combinations of neurons and synapses are described in the following subsections.

1) IF neuron and instantaneous synapse (IF+inst): The dynamics of this model is described by the following equation (setting aside the reset):

$$\frac{dV_i}{dt}(t) = \sum_j \sum_r w_{ij}\delta(t - t_j^r) \tag{3}$$

$V_i(t)$ is the membrane potential of neuron i at time t, w_{ij} is the synaptic weight between neuron i and neuron j, t_j^r is the time of the r^{th} spike from neuron j and δ is the Dirac delta function. For each incoming spike from neuron j, we must read the associated weight w_{ij} and the membrane potential (state) of the neuron. We do not need to read the input value as it is a spike and thus it communicates directly the addresses of the corresponding weight and neuron state. Then the weight is simply added to the state with an AC operation. We assume that, in the event the spikes must be buffered, this energy is included in the communication, which is ignored here. In SNNs, a synapse can receive several spikes. Hence, the energy associated to a SNN synapse must be multiplied by the average number of spikes received per synapse ($N_{spikes/syn}$). We obtain the total energy:

$$E_{IF+inst} = N_{syn} \times N_{spikes/syn} \times (ER_{weight} + ER_{state} + EW_{state} + E_{AC}) \tag{4}$$

2) LIF neuron and instantaneous synapse (LIF+inst): Its dynamics (without the reset) is described as:

$$\tau_m \frac{dV_i}{dt}(t) = -V_i(t) + \sum_j \sum_r w_{ij}\delta(t - t_j^r) \tag{5}$$

This corresponds to an exponentially decaying membrane potential with time constant τ_m. The evolution of the membrane potential with time can be also described in an iterative formulation:

$$V_i^t = V_i^{t-1} \times exp(-\frac{1}{\tau_m}) + \sum_j w_{ij}\epsilon_j^t \tag{6}$$

In this model, the states of neurons are updated at each timestep. Therefore, in equation (4) we must add the energy for updating the state for all neurons (N_{neur}) and all timesteps (T is the number of timesteps in one inference), corresponding to: reading the state, multiplying it with a constant, and writing back the result. We obtain the total energy by combining the

operations performed at each incoming spike, from (4), and at each timestep:

$$\begin{aligned} E_{LIF+inst} = N_{syn} \times N_{spikes/syn} \times (ER_{weight} \\ +ER_{state} + EW_{state} + E_{AC}) \\ +N_{neur} \times T \times (ER_{state} + EW_{state} + E_{MAC}) \end{aligned} \quad (7)$$

Another strategy is to update the states only when necessary, i.e. when there is an incoming spike to the neuron (using an additional variable to record the last spike time), as proposed in [25]. However we found that in recent SNN algorithms, the update at each timestep is more efficient, as $N_{syn} \times N_{spikes/syn}$ is large compared to $N_{neur} \times T$, as will be seen in Section II-E. Moreover, the energy of updating the states when there is an incoming spike is higher than updating at a timestep (as in addition we must compute the total decay from the last to the current input spike).

3) IF neuron and continuous synapse (IF+cont): SNNs with temporal coding, for instance Time-to-First-Spike (TTFS) coding, require continuous synapses to track the spike timings [7]. The equations of the *IF+cont* model (without the reset) are:

$$\begin{aligned} \frac{dV_i}{dt}(t) &= I_i(t) \\ \tau_s \frac{dI_i}{dt}(t) &= -I_i(t) + \sum_j \sum_r w_{ij} \delta(t - t_j^r) \end{aligned} \quad (8)$$

$I_i(t)$ is the input current state variable of neuron i at time t. This corresponds to an exponentially decaying synapse current with time constant τ_s. The neuron is IF because the membrane potential only integrates the input current and does not decay over time. We use the discretized formulation in [7]:

$$\begin{aligned} V_i^t &= V_i^{t-1} + I_i^t \times \lambda \\ I_i^t &= I_i^{t-1} + \sum_j w_{ij} \epsilon_j^t - I_i^{t-1} \times \lambda \end{aligned} \quad (9)$$

Note that the equation of the input current I is similar to the one of the membrane potential V for the *LIF+inst* model, using the constant $\lambda \approx \frac{1}{\tau_s}$ to represent the temporal resolution. To update I at each timestep, we must read I, multiply it by a constant, and write back I. To update V at each timestep, we must read V, multiply the previously obtained value of I by a constant and add it to V, and write back V. The update at each spike (instead of each timestep) is not efficient for this model due to the continuous synapse. Indeed, an input spike can generate an output spike even after the input spike time, as the membrane potential keeps integrating the continuous postsynaptic potential. Therefore, keeping track of the spike times is highly complex. We obtain the total energy:

$$\begin{aligned} E_{IF+cont} = N_{syn} \times N_{spikes/syn} \times (ER_{weight} + ER_I \\ +EW_I + E_{AC}) + N_{neur} \times T \times (ER_I \\ +EW_I + ER_{state} + EW_{state} + 2 \times E_{MAC}) \end{aligned} \quad (10)$$

Finally, compared to the *IF+cont* model, the LIF neurons with continuous synapses combination (*LIF+cont*) adds only a MAC operation at each timestep (corresponding to the multiplication of the membrane potential with the decay factor).

TABLE I
NORMALIZED ENERGY COST RELATIVE TO A MAC OPERATION.

CMOS Techno. (data precision)	E_{AC}	E_{MAC}	ER/W	ER/W^{reg}
45nm (8 bit) [14] (Section II)	0.13x	1x	5.4x	/
65nm (16 bit) [26] (Section III)	0.06x	1x	6x	1x

D. Comparison of SNN models

We now compare the energy efficiency of the different SNN models described in the previous subsection. We choose 8 bit Fixed Point data format as it is commonly used for inference in DNN accelerators [10]. We consider a spatially expanded architecture for both ANNs and SNNs using SRAMs as on-chip memory, with energy ratios for compute and memory in CMOS 45nm from [14] (see Table I). For *LIF+inst* and *IF+cont* models, for which there is an energy associated with both neuron and synapse, we assume $N_{spikes/syn} = 1$, we consider $T = 10$ and $T = 500$, and we use the VGG16 topology to compute N_{syn} and N_{neur}.

The relative energy consumption of the memory and compute associated with synapse operations (at each spike) and neuron operations (at each timestep) of the different SNN models is presented in Fig. 2. We observe that, in all models, the energy cost of compute is very small compared to the one of memory. In SNN *IF+inst* compared to ANN, the energy associated with computation is smaller due to the replacement of the MAC by the AC operation. For the *LIF+inst* and *IF+cont* models, the overhead associated with the updates of neurons at each timestep is negligible if the number of timesteps is small (T = 10) but becomes important if the number of timesteps grows (T = 500). In the latter, the energy of updating neurons reaches 15.17% (respectively 26.34%) of the total energy consumption of the *LIF+inst* (respectively *IF+cont*) model.

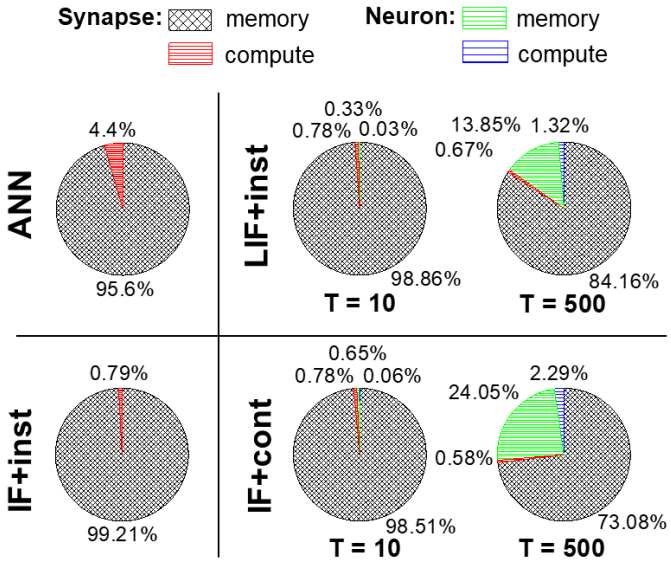

Fig. 2. Relative energy consumption of the memory and compute associated with synapse operations (at each spike) and neuron operations (at each timestep) of the ANN and the different SNN models.

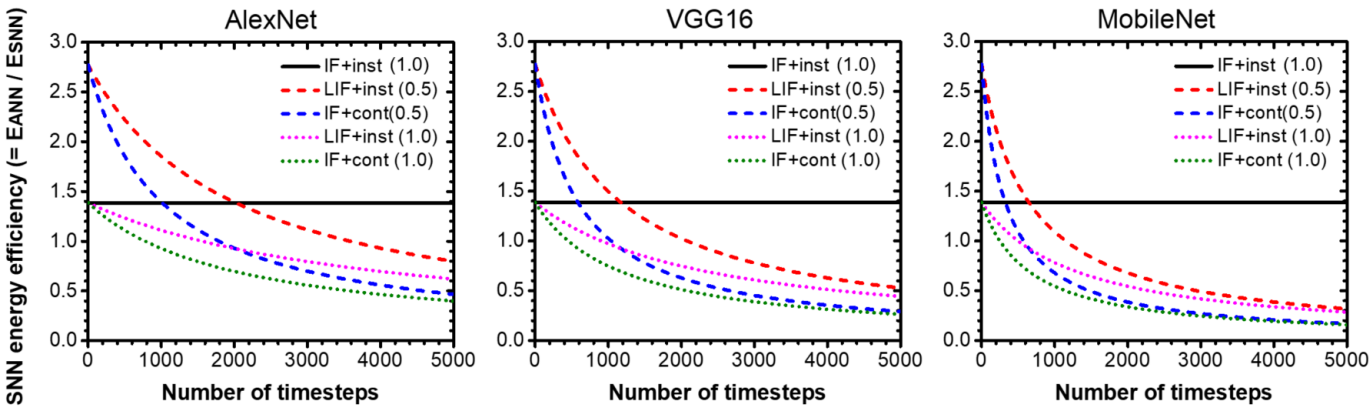

Fig. 3. SNN energy efficiency relative to ANN (=E_{ANN}/E_{SNN}) as a function of the number of timesteps (T), depending on the SNN model and $N_{spikes/syn}$ (in parenthesis). The AlexNet, VGG16 and MobileNet topologies have N_{syn}/N_{neur} = 2.9x10^3, 1.7x10^3, 9.4x10^2, respectively.

TABLE II
SOA SNN ENERGY EFFICIENCY RELATIVE TO ANN (= E_{ANN}/E_{SNN}) USING MODELS AND ENERGY RATIO IN TABLE I FROM SECTION II. NO DATA REUSE AND SPARSITY BENEFITS ARE CONSIDERED FOR ANN.

Task	Paper (Topology)	SNN model	Acc. (%)	T	$N_{spikes/syn}$	Energy efficiency
CIFAR-10	[13] (VGG8*)	IF+inst	90.98	-	0.30	**4.6x**
	[27] (VGG16)	IF+inst	90.35	-	1.30	**1.1x**
	[28] (VGG16)	IF+inst	92.79	-	0.51	**2.7x**
	[11] (ResNet11)	LIF+inst	90.95	100	3.60	**0.4x**
	[29] (VGG9)	LIF+inst	90.50	25	0.80	**1.7x**
	[12] (VGG16*)	LIF+inst	92.70	5	0.39	**3.6x**
	[30] (VGG16)	IF+cont	92.68	680**	0.62	**1.3x**
ImageNet	[27] (VGG16)	IF+inst	68.93	-	5.00	**0.3x**
	[28] (VGG16)	IF+inst	72.59	-	1.00	**1.4x**
	[12] (VGG16*)	LIF+inst	69.00	5	0.41	**3.4x**

* with encoding layer. ** assumed number of timesteps.

In addition, the relative energy efficiency of SNN models with updates at each timestep compared to ANNs also depends on the ratio N_{syn}/N_{neur}. This ratio depends on the network topology. For instance, in the more recent MobileNet topology, this ratio is 3x lower than in the AlexNet topology. The energy efficiency of the different SNN models compared to ANNs (= E_{ANN}/E_{SNN}) depending on the number of timesteps is shown in Fig. 3, for a given $N_{spikes/syn}$. The results are shown for the AlexNet, VGG16 and MobileNet topologies, having N_{syn}/N_{neur} = 2.9x10^3, 1.7x10^3, 9.4x10^2, respectively. We see that, in topologies with a low ratio N_{syn}/N_{neur} (such as MobileNet), the energy efficiency of the *LIF+inst* and *IF+cont* models decreases more rapidly with the number of timesteps compared to topolgies with a higher ratio (such as VGG16 or AlexNet).

E. Application to SNN algorithms

We apply the models previously described to investigate the energy efficiency of state-of-the-art SNN algorithms compared to an ANN (see Table I for the energy ratios used). $N_{spikes/syn}$ in a given layer is the average number of spikes fired by a neuron in the previous layer. To obtain an average on the entire network, we need to weight it by the number of synapses in each layer, which is the number of neurons in this layer multiplied by its fan-in (number of input connections). In practice, if the average number of spikes per neuron of each layer is not available, we assume that over the entire network $N_{spikes/syn} \approx N_{spikes/neur}$ (removing the contribution of the fan-in). Note that some SNN papers use an encoding layer (the first layer receives real pixel values instead of spikes and therefore does MAC operations as in an ANN) to decrease the number of spikes per inference [12], [13]. In that case, only the energy of spiking layers is considered. For the paper [30] using a temporal coding, no temporal resolution is given. Hence, it is assumed based on the paper [31] using a similar SNN model with temporal coding, achieving comparable accuracy with the same network topology and dataset.

For the *IF+inst* model, $N_{spikes/syn}$ must be lower than 1.38 for E_{SNN} to be lower than E_{ANN} regardless of the network topology. To compare the other SNN models, which requires an update at each timestep, we must take into account the network topology and the temporal resolution. The energy efficiency of SNN relative to ANN (= E_{ANN}/E_{SNN}) for state-of-the art SNN papers is shown in Table II (using the naïve ANN implementation described in this section for comparison and the energy ratios from Table I).

We observe that all SNN algorithms have a higher energy efficiency than the corresponding ANN (up to 4.6x more energy-efficient), except in [11], [27] (where $N_{spikes/syn}$ is higher). As in most cases $N_{syn} \times N_{spikes/syn}$ is large compared to $N_{neur} \times T$, the energy of updates at each timestep becomes negligible compared to the energy of synaptic operations. In

that case, all SNN models have a similar energy consumption (similar to the one of *IF+inst*) and only $N_{spikes/syn}$ determines the energy efficiency compared to the ANN. This will not be the case if the number of timesteps increases, or with a different network topology where the ratio between synapses and neurons is smaller, as shown previously in Fig. 2 and 3. Note that even if in these examples, all SNN models have a similar energy consumption, SNN models with an update at each timestep require a time discretization of the inference and the computations are not fully event-based. Moreover, the continuous synapse introduces another state variable for each neuron to store the input current, increasing the memory requirements. In addition, we did not observe a higher accuracy or a higher spike sparsity in the models with leaky neurons or continuous synapses, which could justify their use. For all these reasons, the *IF+inst* model seems a better choice for a digital SNN implementation.

In an event-based SNN implementation, no data reuse is possible (due to the non-flexible and non-predictable computations) and spike sparsity is leveraged naturally (due to the event-driven computations). In comparison, we considered a naïve ANN implementation (worst case ANN) which does not leverage sparsity and data reuse. Therefore, in the next section, we will consider more favorable ANN models. **In the next section, only the *IF+inst* model is considered**, as it is the more general, and the target SNN sparsity can be computed independently of the network topology.

III. ANN MODELS CONSIDERING DATA REUSE AND EXPLOITATION OF SPARSITY

In the previous section, we ignored the opportunities to exploit data reuse and sparsity in ANNs, although they improve the energy efficiency. Data reuse (for all kind of data types: weight, *iact* and *psum*) is the number of times a data that has been read once from a distant memory can be reused locally for a MAC operation. The ideal (theoretical limit) data reuse is that each data is only read once from a distant memory and then reused locally in the PEs. In practice, due to hardware constraints, the reuse is never ideal but is optimized with the dataflow. On the other hand, sparsity can be exploited in *iact* and weights, by gating or skipping unnecessary MAC operations (i.e. with a zero operand), and compressing data. In SNNs, *iacts* are already compressed (only non-zero *iacts*, i.e. spikes, are transmitted) and thus the read of weights and AC are only performed when there is a spike. In ANNs, it requires more logic to process compressed *iact*. Exploitation of sparsity in the weights is not taken into account in this study, as we assume that ANNs and SNNs can process sparse weights and obtain the same benefits. Note that it may be even easier for SNNs to process sparse weights than for ANNs to process both sparse weights and *iacts*. Indeed, the logic would only consist in checking if one operand is zero (skipping of zero spikes is natural), while in an ANN, it has to find the match between two non-zero operands [9]. In this section, we use the energy ratios for memory and compute for CMOS 65nm and the 16 bit Fixed Point data format from the Eyeriss paper [26] (see Table I).

A. Best case ANN: ideal exploitation of data reuse and input activation sparsity

We first investigate the best case for the ANN, corresponding to an optimal data reuse and exploitation of sparsity in the *iacts*. This model gives an upper bound on the ANN energy efficiency relative to the SNN *IF+inst* model, which is independent of the hardware architecture.

Theoretical data reuse, or Reuse Factor (RF), is computed for each layer of a topology given its shape and size. In a fully connected layer, the RF on *iacts* is the number of output neurons, the RF on *psums* is the number of input neurons and there is no reuse of weights (RF=1). In a convolutional layer, the RFs depends on the number of input and output channels, kernel size, image size and stride, details of the formula are given in [32]. The RF of each data types for each layer of AlexNet, VGG16 and MobileNet are shown in Fig 4. We observe that theoretical RFs are very high, on the order of 10^2 to 10^3 on average. In recent architectures, such as MobileNet (using depth-wise convolution), there is fewer reuse for every data type.

Compared to the previously used ANN energy equation (2), we weight the access to a distant memory by the corresponding RF. To reuse data locally, data must be stored in a local storage (such as register files) in the PEs, whose access energy is reduced compared to the distant memory. Then, a data can be accessed from a PE (in the PE or in a neighbor PE from the PEs array) each time it is reused for a MAC. We add this local storage access (ER^{reg}, EW^{reg}) in the ANN energy:

$$\begin{aligned} E_{ANN(reuse)} = N_{syn} \times (&\frac{ER_{iact}}{RF_{iact}} + \frac{ER_{weight}}{RF_{weight}} \\ &+ \frac{ER_{psum} + EW_{psum}}{RF_{psum}} + ER^{reg}_{iact} \\ &+ ER^{reg}_{psum} + ER^{reg}_{weight} + EW^{reg}_{psum} + E_{MAC}) \end{aligned} \tag{11}$$

This equation gives the energy efficiency of the ANN exploiting the maximum data reuse but not sparsity, if we must store and access each data in a local storage in the PEs. In that case, the SNN target sparsity to obtain the same energy efficiency as the corresponding ANN is $N_{spikes/syn} = 0.28$. This is much lower than the target sparsity obtained with the naïve ANN implementation previously described (1.38).

We now consider the ideal exploitation of sparsity in the *iacts*. When there is a zero *iact*, the MAC, the weight read and *psum* read and write in the local memory are saved:

$$\begin{aligned} E_{ANN(reuse+sparsity)} = N_{syn} \times (&\frac{ER_{iact}}{RF_{iact}} + \frac{ER_{weight}}{RF_{weight}} \\ &+ \frac{ER_{psum} + EW_{psum}}{RF_{psum}} + ER^{reg}_{iact} + (1-\gamma) \times (E_{MAC} \\ &+ ER^{reg}_{psum} + ER^{reg}_{weight} + EW^{reg}_{psum})) \end{aligned} \tag{12}$$

γ is the average rate of zero in *iacts*, which is 58% in convolutional layers of AlexNet and VGG16 [18]. We did not consider data compression, which can further reduce the energy consumption by reducing the energy of distant memory accesses, as the distant memory accesses are already negligible due to the ideal RFs. Using this equation, the SNN target

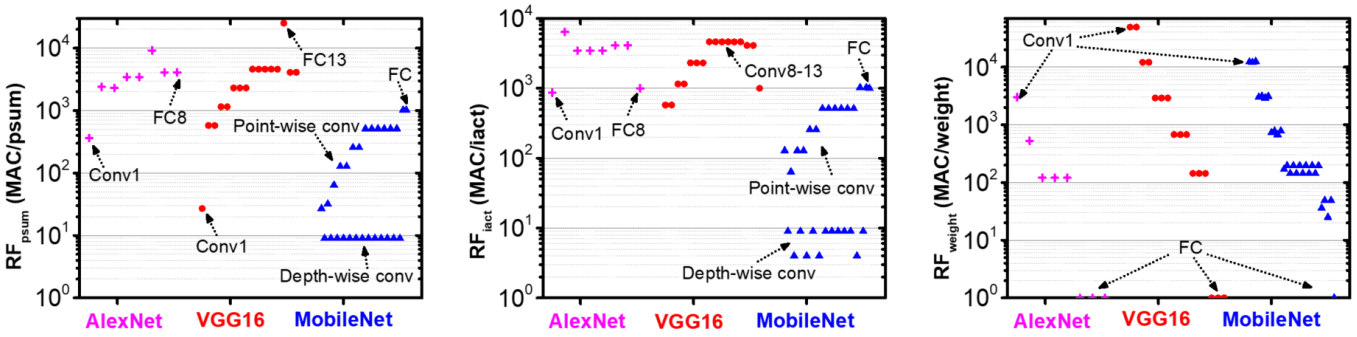

Fig. 4. Ideal data reuse (Reuse Factor) of the three data types (left: *psum*, middle: *iact*, right: weight) for AlexNet, VGG16 and MobileNet topologies (inspired by [10]). Each point represents a layer of the DNN.

sparsity becomes $N_{spikes/syn} = 0.15$. This target sparsity is very low and not achieved in current SNN algorithms as shown in Table II-D. The target sparsity for the SNN *IF+inst* model to be at least as efficient as the ANN, as a function of the average RF (of all data types) in the ANN (assuming 58% zero *iacts*) is shown in Fig. 5. We observe that the target sparsity decreases rapidly with the ANN RF and becomes very small (lower than 0.3) from $RF = 10$.

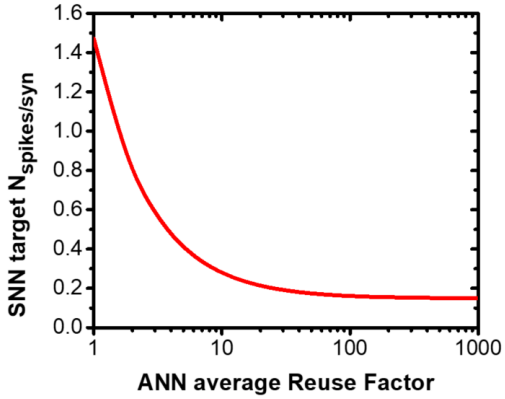

Fig. 5. Target $N_{spikes/syn}$ for the SNN *IF+inst* model to be at least as efficient as an ANN with ideal data reuse and *iact* exploitation of sparsity, as a function of the average data Reuse Factor in the ANN.

In practice, such RFs are not achieved due to hardware constraints and exploitation of sparsity requires additional logic consuming energy. Therefore, in the following subsection, we will consider the case of the Eyeriss v1 and v2 accelerators [10], [18].

B. Real case study: the Eyeriss accelerator

Eyeriss is representative of state-of-the-art DNN accelerators with high energy efficiency leveraging both data reuse and sparsity. This allows us to compare the SNN energy efficiency with a realistic ANN hardware implementation.

1) Eyeriss v1: Eyeriss [18] use a Row Stationary dataflow to increase the data reuse. We use the number of memory accesses given in [18] to compute the actual RFs. *Iacts*, *oacts* and weights are transferred between the DRAM and PEs through the Global Buffer (GLB), while *psums* are only stored in the GLB. The weights are stored in SRAM in the PEs, which means that they are probably read once in the DRAM and then once in the GLB, to be stored in the PEs. Therefore, we can consider only the number of GLB accesses (as there is no off-chip memory in the spatially expanded architecture). We remove the weights accesses to compute the RFs of *iacts* and *psums*. For this purpose, we compare the number of GLB accesses with the number of accesses required without reuse, as in the naïve implementation described in the previous section (each MAC operations requires 4 memory accesses).

We compute the RFs for the AlexNet and VGG16 topologies implemented in [18] for the ImageNet dataset. For the convolutional layers of VGG16 with batch size of 3, 46.04G MACs are performed in total. This would require 276GB *psums* and *iacts* accesses without reuse (data are encoded in 2B). Instead they perform 11006MB GLB accesses. Therefore the effective average RF for *iacts* and *psums* is 25. Similarly for Alexnet convolutional layers, we obtain an average RF for *iacts* and *psums* of 80. They consider only convolutional layers, which consumes more energy compared to fully connected layers in DNNs. They use a batch size superior to 1 to increase the weight reuse, and we consider a batch size of 1, but our computation remains the same (as this does not impact the *psums* and *iacts* reuse). In addition to *iacts* and *psums* accesses in a distant memory, we must consider local data accesses, due to reuse, from local register files which can be either in the PE (x1 energy cost compared to a MAC) or in neighbor PEs in the PE array (x2). To simplify, we assume they are accessed in the PE.

Eyeriss leverages *iact* sparsity with data gating logic in the PEs (MAC and weight read are gated when *iact* is zero), which can save 45% of the PEs power consumption. We interpret this as the following: when there is a zero *iact*, the power consumption of PEs is only 55% of the power consumption when the *iact* is non-zero, where the power consumption of the PE corresponds to the associated MAC operation and memory accesses. The operation is gated but hardware cycles are still spent semi idle. Thus, the obtained energy for a gated operation is only affected by the power as time is constant. This saving includes static energy consumption, which was not considered in our model. As we do not know the relative consumption of static and dynamic factors in Eyeriss PEs, we assume the 45% energy savings corresponds to the savings in

the dynamic energy considered here. We obtain the energy of the ANN Eyeriss model:

$$E_{ANN(Eyeriss)} = N_{syn} \times ((1-\gamma) + 0.55 \times \gamma) \times (ER_{weight} + \frac{ER_{iact} + ER_{psum} + EW_{psum}}{RF_{avg}} + E_{MAC} + ER^{reg}_{iact} + ER^{reg}_{psum} + EW^{reg}_{psum}) \quad (13)$$

$\gamma = 0.58$ is the average rate of zero *iact* in AlexNet and VGG16. RF_{avg} is the average RF computed for *iacts* and *psums*, which is 25 (resp. 80) for VGG16 (resp. Alexnet). Note that the cost to read weights once from the buffer before storing them in the PEs does not appear in the equation. Indeed, we assume that the associated energy is negligible as it corresponds to the maximum reuse of weights. However, weights are stored in SRAM in the PEs, thus the energy cost to read them is the same as the cost to read a data in the buffer. Comparing the total energy of the ANN Eyeriss and SNN *IF+inst* models, we get that $N_{spikes/syn}$ in the SNN must be lower than 0.44 (resp. 0.42) for VGG16 (resp. AlexNet) topology, for the SNN to be more energy-efficient than the ANN. This target spike sparsity is much lower than the one corresponding to the naive ANN implementation (1.38), but higher than the one corresponding to the ideal best case ANN (0.15).

2) Eyeriss v2: Eyeriss v2 [10] is more energy-efficient than the v1 due to a flexible hierarchichal mesh on-chip network (NoC) and better sparsity exploitation in PEs. Both *iacts* and weights are compressed and processed by the PEs directly in the compressed form. Therefore, MACs with zero weight or *iact* are skipped (and not gated as in v1). Pruned networks are used to increase the sparsity in weights, improving the energy benefits. Compared to v1, the Eyeriss v2 achieves x3.0 (resp. x1.9) higher energy efficiency with AlexNet (resp. MobileNet v1). The benefits are higher when using a pruned version of the networks (x11.3 and x2.5, respectively).

However, the paper lacks some important metrics for us to compute the corresponding energy equation, for instance GLB accesses are not specified. Therefore, we use the comparison between the two versions, given in the Eyeriss v2 paper, to translate it into the comparison between SNN and ANN with Eyeriss v2 model. We assume that SNNs can exploit the sparsity in weights of pruned network topologies with the same energy benefits, as explained at the beginning of the Section III. Thus, it is fair to compare the previous results on SNNs with the Eyeriss v2 without using pruned networks. Moreover, we did not consider the NoC efficiency in this study. Therefore, if we consider only the benefits due to sparse PEs with non-pruned networks, the energy efficiency is only improved by x1.15 (x1.06) for the AlexNet (MobileNet) topology from the v1 to the v2. Indeed, the *iacts* sparsity alone compensate slightly the overhead of the sparse PEs logic and the compression of non-sparse data. Therefore, we consider that the target sparsity in the SNN must be x1.15 (for the AlexNet topology) lower compared to the one obtained considering Eyeriss v1, and thus becomes $N_{spikes/syn} = 0.37$.

C. Summary of the results

The relative energy consumption of the local memory, MAC and distant memory in the baseline, ideal reuse + *iact* sparsity and Eyeriss v1 ANN models is depicted in Fig. 6. We observe that in the baseline model (worst case), all the memory consumption comes from a distant memory (96% of the total energy consumption), while in the ideal reuse + *iact* sparsity model (best case), it comes from a local memory (84.06% of the total energy consumption, against 0.32% from memory accesses in a distant memory). Eyeriss v1 is closer to the best case than the worst case, with only 2.2% of the energy consumption due to a distant memory access and 88.02% to a local memory, showing the effectiveness of the dataflow to optimize data reuse.

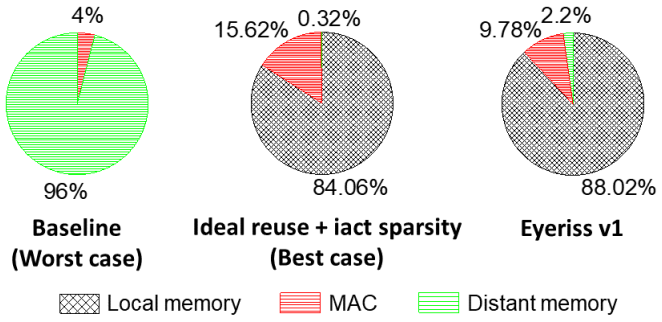

Fig. 6. Relative energy consumption of the local memory, MAC and distant memory in the three ANN models (described in equations (2), (12) and (13), from left to right). AlexNet topology is used in Eyeriss v1 case.

The main results of this paper for evaluating the energy efficiency of SNN *IF+inst* model compared to the previously described ANN models are summarized in Fig. 7 and Table III. Fig. 7 shows the SNN *IF+inst* energy efficiency relative to the ANN (=E_{ANN}/E_{SNN}) as a function of $N_{spikes/syn}$ for each ANN model (using AlexNet topology for Eyeriss v1 and v2). Table III gives the target sparsity for the SNN *IF+inst* model to have the same energy efficiency as the ANN. This corresponds in Fig. 7 to the value $N_{spikes/syn}$ in the x-axis of the intersection between the y-axis at SNN energy efficiency = 1 and the curves. We see that above 0.5 spikes per synapse per inference, SNNs can not compete with ANNs in the realistic (Eyeriss) and ideal cases. However, the SNN energy efficiency grows rapidly as $N_{spikes/syn}$ decreases. For instance, with a spike sparsity of 0.1, the SNN is 3.6x (resp. 1.5x) more energy-efficient than the ANN implementation of the Eyeriss v2 model (resp. the ideal model), and 7.3x (resp. 3.0x) if the spike sparsity is 0.05.

IV. Hybrid ANN-SNN implementations

The results show that SNNs energy efficiency compared to ANNs mainly depends on the SNN spike sparsity. However, we considered the average SNN spike sparsity at the network level, although spike activity is very different from one layer to another. The SNN layer-wise spike activity depends on various factors such as the training methodology, DNN topology and dataset, as shown in Fig. 8. A frequent pattern observed is that spike activity decreases with the depth of the layer.

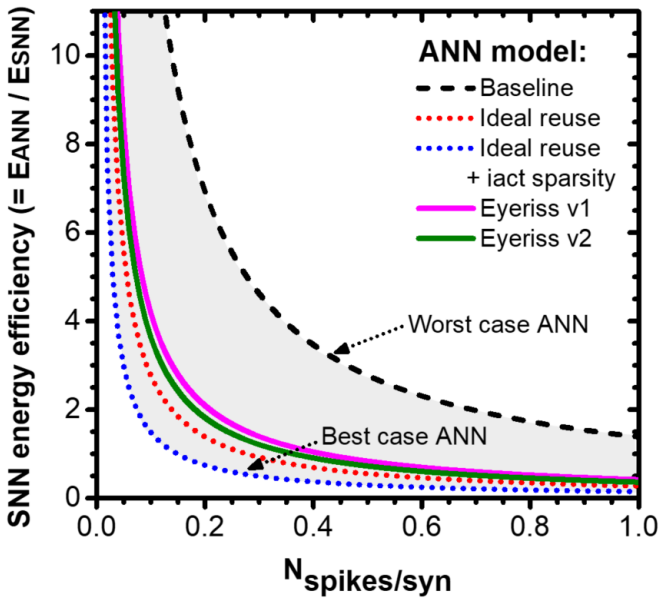

Fig. 7. SNN *IF+inst* energy efficiency relative to ANN (=E_{ANN}/E_{SNN}) as a function of $N_{spikes/syn}$ for the different ANN models considered, using AlexNet topology in Eyeriss v1 and v2 models.

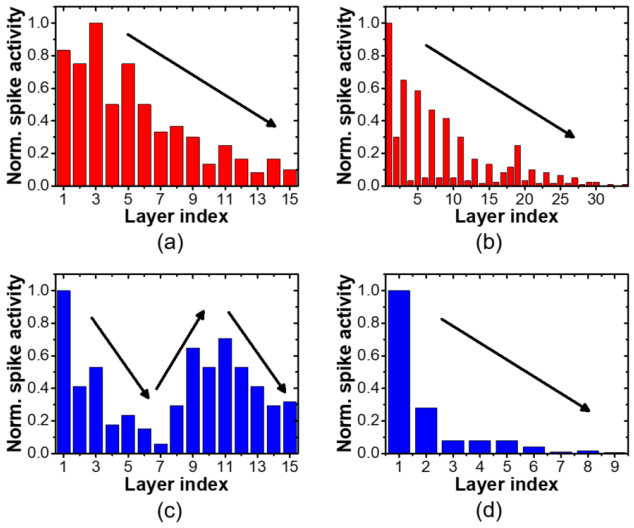

Fig. 8. Normalized layer-wise spike activity of different DNNs with different training methods and datasets. (a) VGG16 from conversion pre-training with spiking backpropagation fine-tuning [12] on ImageNet. (b) ResNet34 from conversion [33] on ImageNet. (c) VGG16 from conversion pre-training with spiking backpropagation fine-tuning [12] on CIFAR10. (d) VGG9 with spiking backpropagation training [11] on CIFAR10.

TABLE III
TARGET SPARSITY FOR THE SNN *IF+inst* MODEL TO BE AT LEAST AS EFFICIENT AS THE ANN

ANN accelerator model (Section)	$N_{spikes/syn}$
Worst case ANN: Baseline (II)	**1.38**
Ideal reuse (III-A)	0.28
Best case ANN: Ideal reuse + *iact* sparsity (III-A)	**0.15**
Eyeriss v1: reuse + *iact* sparsity (III-B1)	0.42
Eyeriss v2: reuse + *iact* sparsity + data compression (III-B2)	0.37

AlexNet topology (non-pruned) is used in Eyeriss v1 and v2 cases.

Therefore, hybrid ANN-SNN implementations, i.e. a network with ANN and SNN layers, become of interest. A conversion from analog values to spikes is required bewteen the output of an ANN layer and the input of an SNN layer, and vice versa. Therefore, using ANN layers at the beginning of the network, where the spike activity is typically higher, and SNN layers at the end, seems a simple solution to avoid multiple energy consuming conversions. This requires to find the optimal separation between the ANN and the SNN using the SNN spike activity at each layer.

Taking as example the SNN ResNet34 from [33], using its layer-wise spike activity shown in Fig. 8b (but non-normalized), we can compute the potential efficiency of such hybrid ANN-SNN architecture. Note that the spike activity numbers are approximate and therefore the following results are only indicative. With $N_{spikes/syn}$ = 2.4 over the entire ResNet34 network, the ANN implementation is more energy-efficient than the SNN implementation (6.3x for the Eyeriss v2 model). However, in the last 10 layers of the SNN, the sparsity is much higher ($N_{spikes/syn}$ = 0.18). These layers implemented in SNN are 2.1x more energy-efficient than the corresponding layers in an ANN Eyeriss v2 implementation. Therefore, implementing the first 22 layers in an ANN and the last 12 in a SNN would result in a hybrid ANN-SNN implementation 1.2x more energy-efficient than the ANN Eyeriss v2 implementation. Note that we used the ANN reuse factors and *iact* sparsity in Eyeriss for VGG16, as the ResNet34 topology was not implemented. The efficiency of hybrid ANN-SNN architectures increases with SNN algorithms having a lower spike activity. For instance, the SNN implementation of the VGG16 proposed in [12] (shown in Fig. 8a), using an encoding layer, is 1.1x more energy-efficient than the ANN Eyeriss v2 implementation. However, when implementing the first 6 layers in an ANN, the hybrid ANN-SNN implementation would be 1.3x more energy-efficient than the ANN Eyeriss v2 implementation. Indeed, the last 10 layers in SNN implementation are 2.2x more energy-efficient compared to their ANN implementation due to their high spike sparsity ($N_{spikes/syn}$ = 0.21).

However, sparsity and data RFs in ANNs also vary in the layers. Therefore, we must take into account these layer-wise factors to evaluate the efficiency of a hybrid ANN-SNN architecture. In addition, the energy associated with the conversion process must also be considered.

V. CONCLUSION

This study demonstrates that, contrary to previous thinking, the main advantage of new SNNs accelerators compared to ANNs on digital hardware comes primarly from exploiting the sparsity of spikes and not from the replacement of MAC by AC operations. Moreover, the IF neuron and instantaneous synapse model seems a better choice for digital implementation than models with LIF neurons or continuous synapses. Indeed, it requires the lowest memory accesses and does not depend on a time discretization, while having a similar spike sparsity and accuracy. For the first time, a lower and upper bound for the relative energy-efficiency of SNN models compared to

ANN models is provided. These bounds are based on a naïve worst case ANN implementation and a theoretical best case assuming perfect data reuse and exploitation of sparsity. The SNN energy efficiency compared to ANN implementations in Eyeriss v1 and v2 accelerators was also investigated. The results showed that current SNN algorithms do not reach a sufficient spike sparsity at the network level to compete with efficient ANN accelerators such as Eyeriss. Hybrid ANN-SNN architectures with the first layers of the network processed in ANN mode and the last layers in SNN mode, appear to be a promising solution to leverage the best of both worlds. In addition, SNN implementations of compact DNNs, such as MobileNet, which offer fewer opportunities of data reuse, may be particularly relevant.

ACKNOWLEDGEMENTS

The authors thank Adrian Evans and Ivan Miro Panades for their valuable advice. This work has been partially supported by MIAI @ Grenoble Alpes, (ANR-19-P3IA-0003).

REFERENCES

[1] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[2] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[3] A. Esteva *et al.*, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, pp. 115–118.

[4] P. A. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[5] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[6] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A Scalable Multicore Architecture With Heterogeneous Memory Structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 1, pp. 106–122, Feb. 2018.

[7] H. Mostafa, B. U. Pedroni, S. Sheik, and G. Cauwenberghs, "Fast classification using sparsely active spiking networks," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4, iSSN: 2379-447X.

[8] F. Corradi, G. Adriaans, and S. Stuijk, "Gyro: A Digital Spiking Neural Network Architecture for Multi-Sensory Data Analytics," in *Proceedings of the 2021 Drone Systems Engineering and Rapid Simulation and Performance Evaluation: Methods and Tools Proceedings*. Budapest Hungary: ACM, Jan. 2021, pp. 9–15.

[9] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient Processing of Deep Neural Networks," *Synthesis Lectures on Computer Architecture*, vol. 15, no. 2, pp. 1–341, Jun. 2020.

[10] Y.-H. Chen, T.-J. Yang, J. S. Emer, and V. Sze, "Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 292–308, Jun. 2019.

[11] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling Spike-Based Backpropagation for Training Deep Neural Network Architectures," *Frontiers in Neuroscience*, vol. 14, p. 119, Feb. 2020.

[12] N. Rathi and K. Roy, "DIET-SNN: Direct Input Encoding With Leakage and Threshold Optimization in Deep Spiking Neural Networks," *arXiv:2008.03658*, Dec. 2020.

[13] J. Wu, Y. Chua, M. Zhang, G. Li, H. Li, and K. C. Tan, "A Tandem Learning Rule for Effective Training and Rapid Inference of Deep Spiking Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2021.

[14] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014, pp. 10–14.

[15] L. Khacef, N. Abderrahmane, and B. Miramond, "Confronting machine-learning with neuroscience for neuromorphic architectures design," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8.

[16] S. Davidson and S. B. Furber, "Comparison of Artificial and Spiking Neural Networks on Digital Hardware," *Frontiers in Neuroscience*, vol. 15, 2021.

[17] H. Lee, C. Kim, S. Lee, E. Baek, and J. Kim, "An accurate and fair evaluation methodology for SNN-based inferencing with full-stack hardware design space explorations," *Neurocomputing*, vol. 455, pp. 125–138, Sep. 2021.

[18] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[19] K. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 5, pp. 416–434, May 2000.

[20] S. Narayanan, K. Taht, R. Balasubramonian, E. Giacomin, and P.-E. Gaillardon, "SpinalFlow: An Architecture and Dataflow Tailored for Spiking Neural Networks," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. Valencia, Spain: IEEE, May 2020, pp. 349–362.

[21] J.-J. Lee and P. Li, "Reconfigurable Dataflow Optimization for Spatiotemporal Spiking Neural Computation on Systolic Array Accelerators," in *2020 IEEE 38th International Conference on Computer Design (ICCD)*. Hartford, CT, USA: IEEE, Oct. 2020, pp. 57–64.

[22] J.-J. Lee, J. Chen, W. Zhang, and P. Li, "Systolic-Array Spiking Neural Accelerators with Dynamic Heterogeneous Voltage Regulation," in *2021 International Joint Conference on Neural Networks (IJCNN)*. Shenzhen, China: IEEE, Jul. 2021, pp. 1–7.

[23] T.-J. Yang and V. Sze, "Design considerations for efficient deep neural networks on processing-in-memory accelerators," in *2019 IEEE International Electron Devices Meeting (IEDM)*, 2019, pp. 22.1.1–22.1.4.

[24] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2014.

[25] A. Roy, S. Venkataramani, N. Gala, S. Sen, K. Veezhinathan, and A. Raghunathan, "A programmable event-driven architecture for evaluating spiking neural networks," in *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2017, pp. 1–6.

[26] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. Seoul, South Korea: IEEE, Jun. 2016, pp. 367–379.

[27] B. Han, G. Srinivasan, and K. Roy, "RMP-SNN: Residual Membrane Potential Neuron for Enabling Deeper High-Accuracy and Low-Latency Spiking Neural Network," *arXiv:2003.01811*, Apr. 2020.

[28] B. Han and K. Roy, "Deep spiking neural network: Energy efficiency through time based coding," in *Computer Vision – ECCV 2020*. Springer International Publishing, pp. 388–404.

[29] Y. Kim and P. Panda, "Revisiting Batch Normalization for Training Low-latency Deep Spiking Neural Networks from Scratch," *arXiv:2010.01729*, Nov. 2020.

[30] S. Zhou, X. Li, Y. Chen, S. T. Chandrasekaran, and A. Sanyal, "Temporal-Coded Deep Spiking Neural Network with Easy Training and Robust Performance," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, pp. 11 143–11 151, May 2021, number: 12.

[31] S. Park, S. Kim, B. Na, and S. Yoon, "T2FSNN: Deep Spiking Neural Networks with Time-to-first-spike Coding," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, Jul. 2020, pp. 1–6, iSSN: 0738-100X.

[32] R. V. W. Putra, M. A. Hanif, and M. Shafique, "ROMANet: Fine-Grained Reuse-Driven Off-Chip Memory Access Management and Data Organization for Deep Neural Network Accelerators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 4, pp. 702–715, Apr. 2021.

[33] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: VGG and residual architectures," *Frontiers in Neuroscience*, vol. 13, p. 95, 2019.

Manon Dampfhoffer received the M.Sc. degree in Informatics and Applied Mathematics from the Grenoble Institute of Technology, France, in 2019. She is currently working toward the Ph.D. degree at Univ. Grenoble Alpes, France, in the Systems-on-Chip and Advanced Technologies (LSTA) laboratory at CEA LIST and Spintec laboratory, supported by the Multidisciplinary Institute of Artificial Intelligence (MIAI @ Grenoble Alpes). Her current research focuses on models and algorithms for implementing energy-efficient Spiking Neural Networks on neuromorphic hardware at the edge.

Thomas Mesquida joined CEA in 2019, after a PhD in microelectronics. His past research included information encoding within Spiking Neural Network, with a focus on inter-spike interval, its hardware implementations and applications. He is currently pursuing the development lightweight spiking and hybrid neural network implementations for embedded applications, combining memory technology, information encoding and learning method for quantized networks.

Alexandre Valentian joined CEA LETI in 2005, after an MSc and a PhD in microelectronics. His past research activities included design technology co-optimization, promoting the FDSOI technology (notably through his participation in the SOI Academy), 2.5D/3D integration technologies and non-volatile memory technology. He is currently pursuing the development of bio-inspired circuits for AI, combining memory technology, information encoding and dedicated learning methods. Since 2020, he heads the Systems-on-Chip and Advanced Technologies (LSTA) laboratory at CEA LIST. Dr Valentian has authored or co-authored 80 conference and journal papers.

Lorena Anghel received the Ph.D. degree *(cum laude)* from Grenoble INP in 2000. From 2016 to 2020, she was the Vice President of Grenoble INP, in charge of industrial relationships, where she is currently the Scientific Director. She is also a Full Professor with Grenoble INP and a member of the Research Staff of the Spintec Laboratory. She has published more than 130 publications in international conferences and symposia. She has supervised 24 Ph.D. students. Her research interests include hardware design and test of neural networks, on-line testing, fault tolerance, and reliable design and verification. She was a recipient of several best paper and outstanding paper awards. She had fulfilled positions, such as the General Chair and the Program Chair for many prestigious IEEE conferences including the IEEE VTS, the IEEE ETS, and the IEEE On-Line Test Symposium.