IAC–24–D.1.2.11

Investigation of low-energy spiking neural networks based on temporal coding for scene classification

Paolo Lunghi[a*], Stefano Silvestrini[b], Gabriele Meoni[c], Dominik Dold[d], Alexander Hadjiivanov[e], Dario Izzo[f]

[a] *Politecnico di Milano, Aerospace Science and Technology Dept., Via La Masa 34, 20156, Milano, paolo.lunghi@polimi.it*
[b] *Politecnico di Milano, Aerospace Science and Technology Dept., Via La Masa 34, 20156, Milano, stefano.silvestrini@polimi.it*
[c] *ESA–ESTEC, Advanced Concepts Team, Keplerlaan 1, 2201AZ, Noordwijk, gabriele.meoni@esa.int*
[d] *ESA–ESTEC, Advanced Concepts Team, Keplerlaan 1, 2201AZ, Noordwijk, dominik.dold@esa.int*
[e] *ESA–ESTEC, Advanced Concepts Team, Keplerlaan 1, 2201AZ, Noordwijk, alexander.hadjiivanov@esa.int*
[f] *ESA–ESTEC, Advanced Concepts Team, Keplerlaan 1, 2201AZ, Noordwijk, dario.izzo@esa.int*
[*] *Corresponding author*

Abstract

Spiking Neural Networks (SNN) are becoming increasingly interesting to the space domain due to their energy efficiency. Indeed, small satellites use devices with low computational power and have a low system power budget in general. For this reason, SNN represent a promising candidate for the on-board implementation of neural-based algorithms: among those, the scene classification task is of primary importance for the space community and constitutes a valuable test case given the abundance of competitive methods available to establish a benchmark. SNN are also commonly referred to as the third generation of Artificial Neural Networks (ANN), where each neuron in the network uses discrete electrical pulses (spikes) communicating in an event-based manner. SNN have the potential advantage of achieving higher energy efficiency than their ANN counterparts. While generally a loss of accuracy on SNN models is reported, new algorithms and training techniques can help close the gap in terms of accuracy while maintaining a low energy consumption profile. To evaluate the feasibility of applying SNN onboard spacecraft, this research paper focuses on a theoretical analysis and comparison of different SNN techniques applied to the task of scene classification for the EuroSAT dataset. A particular focus is put on information encoding and learning representations for sparse spiking architectures. A widely adopted encoding method is rate-based coding, where information is encoded by the average number of spikes fired by a neuron. Several drawbacks, in particular in the context of low energy applications, have been identified for this method. Instead, temporal coding, where the time signal between spikes drive the encoding, is here investigated due to its inherent sparsity nature and higher efficiency of the resulting spiking network, i.e. lower number of spikes for a given information content. Considerable effort is also dedicated to the evaluation of several competing models, developed following the state-of-the-art in SNN coding (rate and temporal coding) and training techniques, as well as the study of the impact of different neuron and synapse models on the output performance. In addition, this research aims at developing and validating reliable metrics and proxies that can be used to compare different architectures and to establish a clear theoretical dependence between architecture parameters (e.g. number of neurons and spikes) and the expected energy consumption one could expect on-board the spacecraft.

Acronyms/Abbreviations

ANN	Artificial Neural Networks,
BNTT	Batch Normalization Through Time,
CNN	Convolutional Neural Network,
GNC	Guidance, Navigation, and Control,
IFL	non-leaky Integrate-and-Fire Linear,
LCL	Locally Connected Layers,
LIF	Leaky Integrate-and-Fire,
MLP	Multilayer Perceptron,
ROC	Rank Order Coding,
SG	Surrogate Gradient,
SNN	Spiking Neural Networks,
TTFS	Time To First Spike.

1. Introduction

Easier access to orbit, miniaturization and new operative concepts like On Orbit Servicing, Active Debris Removal, and reusability of space assets are disrupting the space industry, as we assist to an unprecedented growth of the number of spacecraft in flight and a renewed impulse to exploration. Nevertheless, the operations of space systems still rely heavily on human intervention and on large, expensive communication infrastructures. These factors are quickly becoming bottlenecks that could seriously hinder the development of this new space renaissance. A high level of autonomy is clearly required by future space systems: Potential applications include, among others, early

detection of potential failures or catastrophic events [1], feature detection and tracking in Guidance, Navigation, and Control [2–5], pre-processing of collected data by Earth Observation satellites to mitigate bandwidth requirements by filtering out invalid or corrupted data, advanced navigation and signal processing tasks [6–8]. On the other hand, space systems are traditionally limited in computational and memory resources, a condition exacerbated by the recent trend towards extreme miniaturization, such as CubeSats and other types of micro and pico satellites [9].

Spiking Neural Networks (SNN) are characterized by low-power and energy-efficient properties [10–18], due to their brain-inspired architecture, in which neurons communicate by means of discrete spikes. Several neuron models exist, but all of them share a general behavior in which input events are somewhat accumulated along time, increasing an internal state called membrane voltage or potential, mimicking the membrane voltage of biological neurons. Whenever the potential exceeds a certain threshold, a spike is released and the voltage is reset. Differently with respect to traditional ANN, computation is performed only at the time a spike is received, making the overall activity inside the network inherently sparse. Their recurrent nature makes SNN particularly suitable to process time series, just as in navigation systems. Moreover, SNN can be implemented on neuromorphic hardware, designed to fully exploit such asynchronous, sparse computation paradigm to achieve solutions capable to outperform those based on ANN in terms of power consumption and energy efficiency [10, 11]. On the other hand, the peculiarities that make SNN so promising make them also harder to train than their ANN counterparts, due to the intrinsically discontinuous nature of the signals flowing inside the network. In fact, no standard training techniques still exist for SNN. Various forms of unsupervised learning were proposed [19–24]. However, an unsupervised learning rule alone is not sufficient for decision making, and they cannot be optimized for a specific target output. Rate-based conversion between deep ANN and SNN demonstrated a minimal loss of accuracy [10, 14, 16, 25]. However, the use of rate-based conversion usually requires high firing rates and a high number of time steps to provide acceptable performance [14, 26], largely reducing the energetic advantages [10, 11, 27, 28]. Methods based on temporal coding, also called *latency-based* methods, might achieve higher efficiency, encoding information in the fire time of neurons [12, 14, 15, 17, 18, 29]. According to the Time-To-First-Spike (TTFS) coding, the more a neuron is activated, the shorter is its firing delay, and more significant the associated transmitted information [12, 14, 15, 18]. For classification tasks, Rank Order Coding (ROC) is even more simple, for the attribution class corresponds to the output neuron that fires first, regardless of the exact spike time. In such encoding schemes, neurons can be even forced to fire once at most during an inference, making SNN models based on temporal coding extremely attractive for energy-constrained applications.

Several development are still needed to achieve practical implementation on real space systems, for neuromorphic research is still a topic in its early development. The aim of this work is to provide some practical tools needed for the design of future spaceborne neuromorphic systems. The primary objective is to estimate the actual gains, in term of trade-off between accuracy and energy, that can be expected from SNN with respect to classical ANN. A novel metric, capable to compare the model complexity of both ANN and SNN in a hardware-agnostic way, is proposed as a proxy for the energy consumption. The performance of several SNN (both rate- and latency-based) and ANN models is compared on a scene classification task, using the EuroSAT RGB dataset [30], a land use classification tasks representative of a practical use case in Earth Observation to reduce bandwidth requirements by detecting invalid images onboard. The selection of a classification task, for which ANN achieves state of the art performances, constitutes the hardest possible benchmark to demonstrate SNN capabilities. In order to provide validation to such approach, the energy trend predicted with the proposed metric is compared with actual measures of energy used by benchmark SNN models running on neuromorphic hardware. The secondary goal is to analyze the internal dynamics of SNN models, identifying the most significant factors which affect the energy consumption, in order to derive design principles useful in future activities.

2. Test methodology

Several models, both SNN and ANN, were trained to assess the impact of different architectures, neuron models, and encoding schemes on the energy consumption. The first trade-off consists in the selection of the training method for the spiking systems. Given the aforementioned limitations of other methods, Surrogate Gradient (SG) was selected as the training method of choice for the possibility to handle different test cases without the need to tailor the entire training for each of them [26, 31–34]. In fact, SG has the advantage of being independent of the specific neuron model and type of encoding, with the ability to leverage traditional deep learning libraries, making it suitable for fast prototyping. In the following, unless explicitly stated otherwise, SG, in its SuperSpike [31] variant, was used as the training method for SNN, leveraging the Norse library [35] for practical implementation.

The second major factor impacting SNN dynamics and

performances is the selection of the proper neuron model [36]. Energy efficiency (rather than biological plausibility) was the main parameter of interest in the adoption of SNN in space applications. Hence, two of the most simple neuron models were considered to minimize computation: the leaky integrate-and-fire (LIF) and the non-leaky integrate-and-fire linear (IFL) neurons. Three main types of network architectures were included in the test cases: Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), and MLP with locally connected layers (MLP/LCL). MLP/LCL is a MLP whose first layer takes 2D tensors (e.g., image data) as input while limiting the receptive field of each neuron in a way that mimics convolutional layers, a possible method to circumvent limitations currently present on some neuromorphic processors which do not support convolution.

2.1 Energy estimation method

The goal of this work is not to develop a method to estimate absolute energy consumption, being the number of the unknown parameters too high for such a task, but rather to achieve a credible way to perform relative comparison among models (both SNN and ANN), useful in trade-offs and during the prototyping phase.

When dealing with ANN running on traditional computing hardware (CPU and GPU), data movement dominates energy consumption at inference, reaching up to 90 % of the total for certain architectures [37, 38]. While the computational energy is directly dependent on the actual number and type of floating point operations, the estimation of the memory energy is more complex, for modern hardware organizes memory in a hierarchical manner, with different speed and cost of access. Hence, even if widely used, the number of floating-point operations and the number of weights in the model might not be ideal metrics for energy computation. It would seem reasonable to adopt a black-box approach as in [39], an attempt to take into account the average contributions of both computational and memory energy. Nevertheless, this approach comes with its own limitations. Machine learning hardware has recently seen an impressive improvement in absolute performance and power efficiency, mostly due to specialized computing units (tensor cores, TPU, etc.). But such performance figures are achieved by means of massive parallelism, achieved by *batch computation* [40]. Then, energy efficiency drops quickly far from nominal values whenever the hardware capacity is not saturated, due to the relatively high power consumption in idle mode. Conversely, the most adopted metrics for SNN are the number of emitted spikes and the number of synaptic operations [41, 42]. Even if a certain increasing trend in energy can be expected with increasing number of spikes, different internal connectivity can lead to a very different number of synaptic operations, making such metric only a very rough estimation. Moreover, such metrics are incompatible with ANN. Some methods have been developed to take into account the memory energy: however, they still require some knowledge about specific hardware features, like the size of the I/O buffer in [43] or the fraction of data reuse in [44].

The number of *Equivalent Multiply and Accumulate Operations (EMAC)* is proposed as a *hardware-agnostic* method suitable for comparing the computational load of both SNN and their ANN counterparts. EMAC is intended as a dimensionless generalization of MAC operations, taking into account both computation and memory costs. In fact, all the floating-point operations performed in standard ANN are of the Multiply and Accumulate (MAC) type, while a significant part of the computation in SNN is made up by Accumulate (AC) operations. Then, different relative weights need to be assigned to AC and MAC in the estimation of the related computational burden. It is here assumed that moving data in memory is the dominant factor in the determination of the energy performance: assuming a unitary weight for MAC ($1\,\text{MAC} = 1\,\text{EMAC}$), a conservative weight value of $2/3$ assigned to the AC ($1\,\text{AC} = 0.667\,\text{EMAC}$), for only 2 numbers are involved in AC, w.r.t. 3 in the MAC. No overhead related to input/output operations, system idle power consumption, or computation overhead due to actual implementation are taken into account. Finally, it is assumed that the network is implemented digitally. Although analog neuromorphic processors promise even lower energy consumption [45–47], their performances are dependent on the specific technological solution, often tailored to a specific neuron model. The assumption of a digital implementation allows a more reliable comparison between different architectures. To simplify the notation. in the following sections the terms *energy consumption* and *computational load* both refer to the value of EMAC required by the network to perform a single inference.

In a spiking neuron, part of the operations (i.e. the *synaptic operations*) are performed whenever a spike is received, while other operations are executed at each time step while updating the neurons internal states. Then, the total number of EMAC operations $\mathrm{E_{tot}}$ depends on two distinct contributions:

$$\mathrm{E_{tot}} = \mathrm{E_{syn}} + \mathrm{E_{upd}} \qquad [1]$$

where $\mathrm{E_{syn}}$ is the synaptic operation contribution, and $\mathrm{E_{upd}}$ is the neuron update. The terms can be further explicited:

$$\mathrm{E_{tot}} = s e_{\mathrm{syn}} + n T e_{\mathrm{upd}} \qquad [2]$$

where s is the total number of synaptic operations per-

formed, n is the number of neurons in the network, and T is the number of time steps. The two terms e_{syn} and e_{upd} are respectively the energy per synaptic operation and the energy per neuron update and they depend on the specific neuron model. The general form of Eq. [2] remains valid even for ANN, in which there is no update in time and connections between layers are synapses with $e_{\text{upd}} = 0$ and $e_{\text{syn}} = 1\,\text{EMAC}$. The procedure to identify the two parameters for a specific neuron model is:

1. Write the neuron's dynamics in its discrete-time form (as in its digital implementation);
2. Identify each operation as MAC or AC;
3. Operations processed at each time instant are related to neuron update and contribute to e_{upd};
4. Operations performed only whenever a spike is received are synaptic operations and contribute to e_{syn}.

The contribution of spike generation/voltage reset is neglected, as is can be considered just a variable assignment operation. As an example, in the following the procedure is shown for the LIF neuron, whose continuous dynamics is represented by the system:

$$\begin{cases} \dfrac{di(t)}{dt} = -\dfrac{i(t)}{\tau_{\text{syn}}} + \sum\limits_{j \in P} w_j S_j(t) \\ \dfrac{dv(t)}{dt} = \dfrac{-v(t) + i(t)}{\tau_{\text{mem}}} - v_{\text{th}} S_{\text{o}}(t) \end{cases} \quad [3]$$

where i is the current, v the potential, and P denotes the domain of the presynaptic neurons. $S_j(t) = \sum_k \delta(t - t_j^k)$ is the input spikes train from the j-th presynaptic neuron, where δ is the Dirac's delta and k is the index of the spikes times. $-v_{\text{th}} S_{\text{o}}(t)$, where $S_{\text{o}}(t) = \delta\big(v_{\text{th}} - v(t)\big)$ and v_{th} is the threshold potential, represents the neuron reset mechanism. Once discretized, it takes the form:

$$\begin{cases} i_{k+1} = i_k - i_k \dfrac{\Delta t}{\tau_{\text{syn}}} + \sum\limits_{j=1}^{n_S} w_j S_{jk} + b \\ v_{k+\frac{1}{2}} = v_k + (i_{k+1} - v_k) \dfrac{\Delta t}{\tau_{\text{mem}}} \\ v_{k+1} = v_{k+\frac{1}{2}} - v_{\text{th}} S_{k+1} \end{cases} \quad [4]$$

taking to the final estimation:

$$\begin{aligned} e_{\text{syn}} &= 1\,\text{AC} = 0.667\,\text{EMAC} \\ e_{\text{upd}} &= 2\,\text{AC} + 2\,\text{MAC} = 3.333\,\text{EMAC} \end{aligned} \quad [5]$$

For the sake of completeness, Eqs. [6] and [7] reports the continuous dynamics and the identified energy parameters for the IFL neuron, also considered in this work.

$$\begin{cases} \dfrac{di(t)}{dt} = \sum\limits_{j \in P} w_j S_j(t) \\ \dfrac{dv(t)}{dt} = i(t) - v_{\text{th}} S_{\text{o}}(t) \end{cases} \quad [6]$$

$$\begin{aligned} e_{\text{syn}} &= 1\,\text{AC} = 0.667\,\text{EMAC} \\ e_{\text{upd}} &= 2\,\text{AC} = 1.333\,\text{EMAC} \end{aligned} \quad [7]$$

The identification of the remaining parameters in Eq. [2] completes the energy estimation. The number of neurons n is immediately known, given the specific network. In case of TTFS/ROC encoding, the latency is determined by the time the first spike is emitted by the output layer. This implies also that the number of time steps T can different in each inference: performances can still be assessed in terms of mean and standard deviation, while for rate-based encoding the time of inference T is known and predefined. The number of synaptic operations s is estimated layer-wise. The number of synaptic operations between a layer l and the previous layer $l-1$ is:

$$s_{(l)} = n_{s(l)}\, n_{n(l)}\, T\, P_{(l-1)}(S) \quad [8]$$

where n_{s} is the number of pre-synaptic connections in each neuron receptive field and n_{n} is the number of neurons in the layer, both of which are dependent on the layer type (convolutional, fully connected etc.). $P_{(l-1)}(S)$ is the probability that a pre-synaptic neuron emits a spike, and it can be estimated as:

$$P_{(l-1)}(S) \sim \frac{N_{s(l-1)}}{T\, n_{n(l-1)}} \quad [9]$$

in which $N_{s(l-1)}$ is the total number of spikes emitted by the previous layer at inference. Substituting [9] into [8], and simplifying the obtained equation, we achieve:

$$s_{(l)} = n_{s(l)}\, n_{n(l)}\, f_{(l-1)} \quad [10]$$

where f_{l-1} is the average spiking rate of the *previous* layer. In case of ANN, $f_{(l-1)} = 1$. In case of recurrent layers, the additional recurrent synaptic operations can be simply computed by:

$$s_{r(l)} = n_{s(l)}\, n_{n(l)}\, f_{(l)} \quad [11]$$

which is similar to Eq. [10] except that the spiking rate *of the same layer* $f_{(l)}$ is used.

2.2 Discussion

The use of EMAC presents some advantages with respect to other, traditionally used, metrics, starting with the possibility to compare SNN with ANN, and the capability to compare different neuron models. At the same time, the discrimination between synaptic operations and neuron updates contributions allows to take into account the impact of the model size, latency, and internal connectivity, indistinguishable by other methods, useful for model energy optimization in design phases. Being hardware-agnostic, it allows relative comparisons between models, enabling at least preliminary information for trade-off. Moreover, the followed approach is intentionally conser-

vative in treating SNN. First, the number of time steps of the model (which impacts significantly the E_{upd} term in Eq. [1]) in SNN is not here optimized. Second, data locality is not considered, assigning the relative weights of MAC/AC operations basing only on the number of quantities involved in the computation. On neuromorphic processors, tailored for asynchronous, parallel computation, weights are stored locally to the single neuron (or to small groups of neurons), further favoring SNN over ANN once on hardware. Any energy consumption contribution but the operations strictly necessary for the inference is not taken into account. Idle power consumption and other possible overhead can have a very large variability depending on the actual hardware platform and should be considered in actual trade-off. Finally, the specific representation of the numerical quantities (i.e. the bit depth) is not considered. Energy consumption can drop dramatically with simpler representations, or by switching to fixed point arithmetic, but it is assumed that model quantization can be performed in both ANN and SNN alike. However, if required EMAC can be easily adapted to include different number formats, simply by proper tailoring of the relative weights attributed to MAC/AC operations.

3. Results

Following the methodology expounded above, 82 test models were trained, with different combinations of neuron models and network architectures. Three neuron models are considered: a) LIF spiking neurons with rate encoding (RATE); b) IFL spiking neurons with Rank Order Coding (ROC), allowed to spike once at most during the inference; c) non-spiking ReLu neurons (ANN). Four general architectures are included: A) Multilayer Perceptrons (MLP) B) MLP architecture with Locally Connected Layer (MLP/LCL); C) Convolutional Neural Networks (CNN); D) MLP with a single recurrent hidden layer (MLP/REC). All the model were trained with SG. The naming convention of the test cases is summarized in Table 1. Several hyperparameters affecting both the model and the training process concur to determine each specific case: only the main ones are synthesized in the naming scheme.

Table 1. Test case naming convention.

ID_ + Type + Architecture + Nlayers + [SpecialProperty]	
ID	Unique numeric ID
Type	**R** (rate coded SNN) | **T** (rank order coded SNN) | **A** (ANN)
Architecture	**M** (Multilayer Perceptron) | **C** (Convolutional)
Nlayers	Number of network layers
SpecialProperty (opt.)	**L** (Locally connected Layer) | **R** (Recurrent) | **B** (BNTT reg.) | **F** (Spike rate reg.)

3.1 Accuracy vs energy

The achieved accuracy vs the energy consumption, shown in Fig. 1 is the main figure of merit in this analysis. Just the cases which attain the best performance for each of the groups mentioned are shown, to highlight a sort of Pareto front of the achievable performance.

SNN are able to reach performance comparable of ANN, with significantly lower energy values. This is particularly evident looking at the latency architectures *45_TC5B*, *49_TC5B*, and *52_TC6B*, corresponding respectively to the ANN cases *74_AC5*, *71_AC5*, and *72_AC6*: a $\sim 2.5\,\%$ drop in efficiency corresponds to a $\sim 60\,\%$ decrease in the EMAC per inference. The same happens for rate-based networks trained from scratch with SG, with the architecture *75_RC4* that hits better performance than latency ones on both accuracy and EMAC. The opposite can be observed for simpler networks: spiking MLP and MLP/LCL present increased accuracy w.r.t. their ANN counterpart. However, for such simple networks, this is achieved at the expense of a larger EMAC value. Moreover, the accuracy of such networks still remains relatively low, under 75 %, a value noncompetitive with convolutional architectures, albeit better than their ANN counterparts. It is worth noting that the number of time steps in the simulation, a hyperparameter not specifically optimized, plays a major role in determining the overall energy performance, especially in MLP in which the low number of neurons forces the information to be compressed, reducing in this way the sparsity of the computation. The Locally Connected Layer, applicable only to the first layer in the network to cope with this specific problem, seems not sufficient to significantly limit the impact of the subsequent fully connected layers. Conversely, convolutional layers seem particularly effective in exploiting the sparsity of the spiking computation.

Fig. 2 compares the EMAC with the most simple metric often adopted in literature to estimate energy, that is the number of spikes emitted at inference. The average values evaluated on the test set are used to build the chart. Even if a general correlation trend is present it can be seen how the use of the number of spikes could lead to large errors with a large variation of computational load among cases with almost the same number of spikes, and vice versa. The reason of this behavior is due to the significant part of

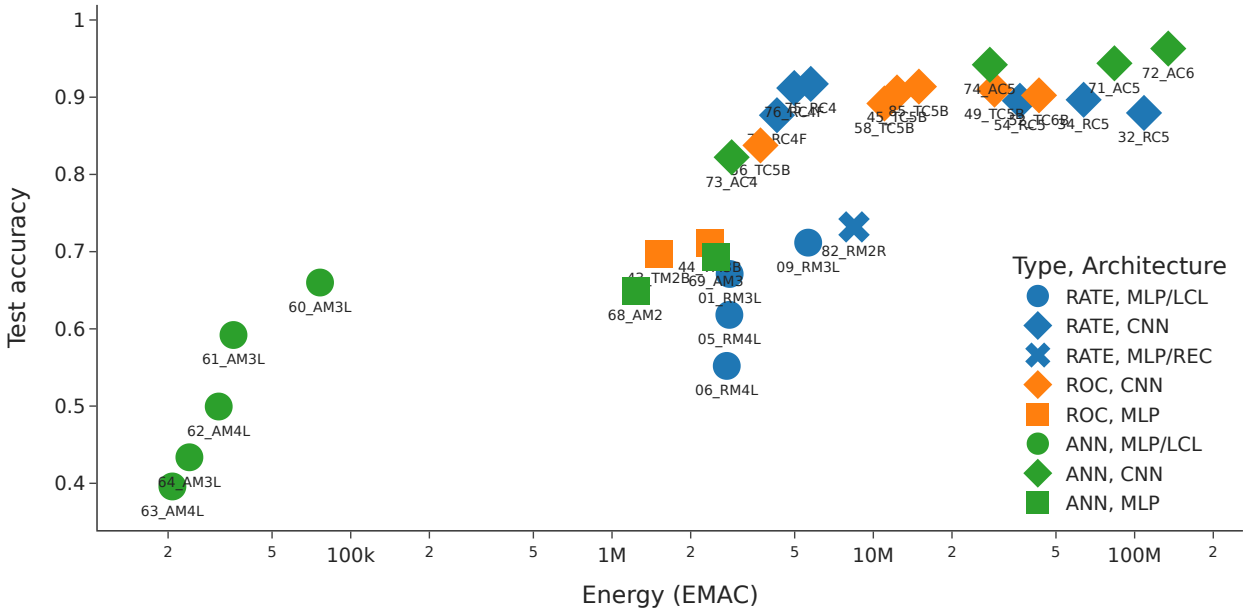


Fig. 1. Accuracy vs dimensionless energy (EMAC).

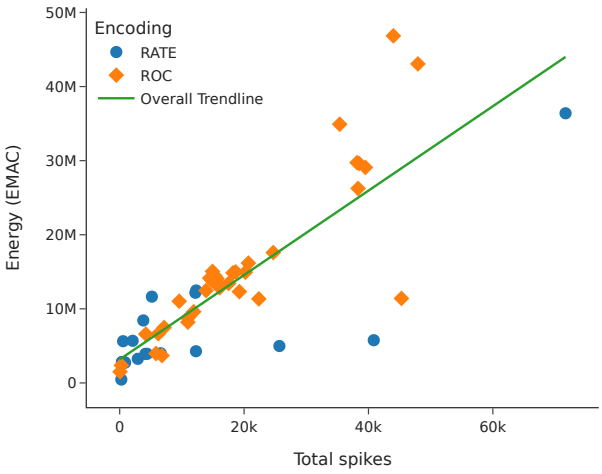


Fig. 2. Dimensionless energy (EMAC per inference) w.r.t. total emitted spikes per inference, average values over the test set.

the energy consumption related to synaptic operations determined by the actual pattern of connection between layers. Fig. 3 shows the breakdown of EMAC among neuron update, synaptic operations, and recurrent synaptic operations for some test cases, while Fig. 4 shows their respective number of emitted spikes. Cases *17_TC5* and *18_TC5* share the same convolutional architecture, in terms of neuron types, latency encoder/decoder, and number and size of spiking layers. The only difference between the twos is the connection pattern between convolutional layers: in *17_TC5* convolution is performed with kernel size $k = 3$ and stride $s = 1$, and dimensional reduction is achieved with standard maxpool layers; conversely, in *18_TC5* convolution is performed with $k = 5$ and $s = 2$, with no maxpooling. Due to the larger receptive field of neurons, *18_TC5* exhibits a $\sim 45\,\%$ increase in energy, even with $\sim 10\,\%$ less spikes. With the same number and type of neurons, and with an almost equal average number of time steps at inference (38.1 ms and 37.1 ms) *17_TC5* and *18_TC5* it can be seen how the increase is completely due to a large difference in the energy spent for synaptic operations. From the energy consumption perspective, it would appear advisable to privilege a low level of internal connectivity. However, it must be said that a larger receptive field in the convolution seems to bring significant benefit to the system performance, with *18_TC5* scoring a $+5.3\,\%$ in the absolute test accuracy w.r.t. *17_TC5*. The discrepancy is even more evident in the case *75_RC4*, still a convolutional architecture. Being a rate-based network, the number of emitted spikes almost doubles the values achieved by *17_TC5* and *18_TC5*. Nevertheless, the lower number of time steps (32 ms instead of 64 ms), and the lower size and number of layers make the average energy almost a third w.r.t. *18_TC5*. *43_TM2B* and *82_RM2R* are another notable example showing the importance of the internal connectivity. They both share the same MLP ar-

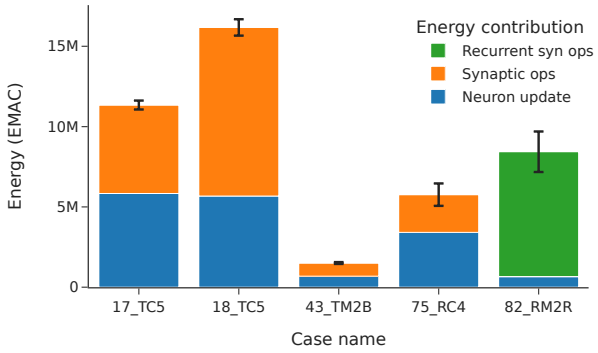


Fig. 3. EMAC per inference for selected test cases: breakdown among network tasks, average value over the test set. Error bars represent the standard deviation.

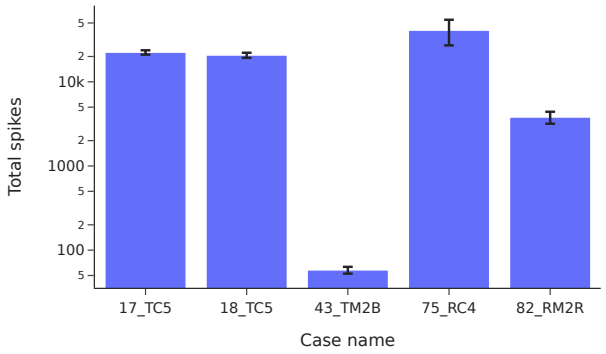


Fig. 4. Total emitted spike per inference, average value over the test set. The standard deviation value is indicated by black error bars.

chitecture with one hidden layer of 100 neurons, but while *43_TM2B* is a spiking MLP with ROC encoding, the hidden layer of *82_RM2R* is made by recurrent LIF neurons with rate encoding. Again the neuron update contribution is practically the same in the two cases. But recurrence involves a higher number of synaptic operations, leading to a total EMAC 5 times higher, despite the higher efficiency in storing information presented by recurrent neurons (reflected by a significant $+3\,\%$ increment in absolute accuracy).

3.2 Effects of regularization

The impact of different regularization methods is shown in Fig. 5. The histograms report test accuracy, EMAC/inference, and the number of emitted spikes per inference for 3 test cases with ROC (*18_TC5*, *9_TC5B*, and *45_TC5B*), and 3 models based on rate encoding (*75_RC4*, *76_RC4F*, *77_RC4F*). For each encoding, the models share the same architecture, with different regularization settings. In the ROC cases, *18_TC5* entails no regularization; neuron-wise Batch Normalization Through Time (BNTT) [42] is applied in *9_TC5B*; spatial BNTT is applied in *45_TC5B*. The number of spikes is similar in all the 3 cases. Neuron-wise BNTT seems beneficial for the energy consumption, but with no improvement on the test accuracy. Spatial BNTT improves both the accuracy, with a 5.4 % gain in the absolute test accuracy, and the energy consumption, with a 24 % drop in EMAC w.r.t. case *18_TC5*. Rate based models are regularized by targeting a predefined spiking rate in each layer at training time. No regularization is applied in *75_RC4*; in *76_RC4F* a target average spiking rate of 2 spikes per neuron per inference was set; the same value was set to 1.5 in *77_RC4F*. In these cases, more restrictive regularization provokes a decrease in the accuracy, with limited improvement in the energy consumption. This phenomenon appears to be connected again to the different sources of energy consumption in SNN. In fact, as shown in Fig. 3, the energy consumption in the *75_RC4* architecture is dominated by neuron update, while a limitation of the spiking rate impacts only the contribution by synaptic operations.

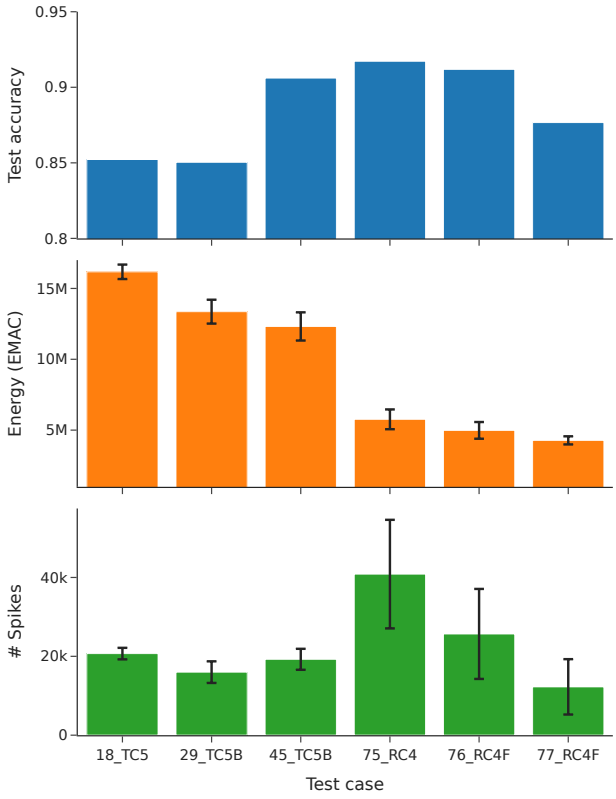


Fig. 5. Effect of regularization. *xx_TC5x* cases make use of Rank Order Coding and BNTT; *7x_RC4x* cases are rate-based with target spiking rate normalization.

3.3 Scaling to deeper architectures

Despite the promising performances achieved in accuracy, SNN still struggle in scaling to deeper architectures. Fig. 6 is a close-up of Fig. 1, limited to the convolutional architectures. Not just the top performing models are shown, but also some other test cases designed to study the trend performances as the deepness increases. All the models in the figure are CNN with number of layers varying from 3 to 6 (denoted by the marker size in the scatter plot). Latency-based (ROC), rate-based (RATE) and their ANN counterparts are included. ANN show a

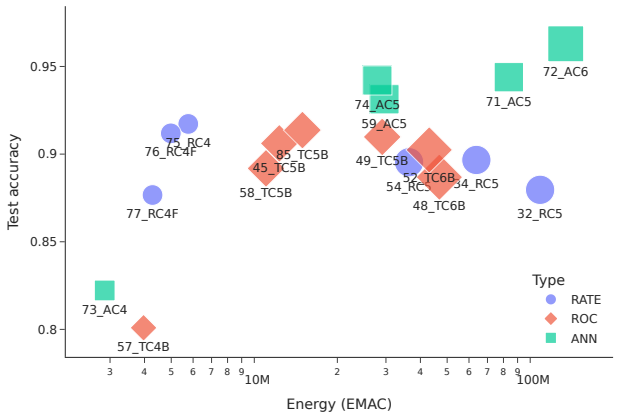


Fig. 6. SNN Scalability. The size of the markers is related to the number of layers in the 3 to 6 range.

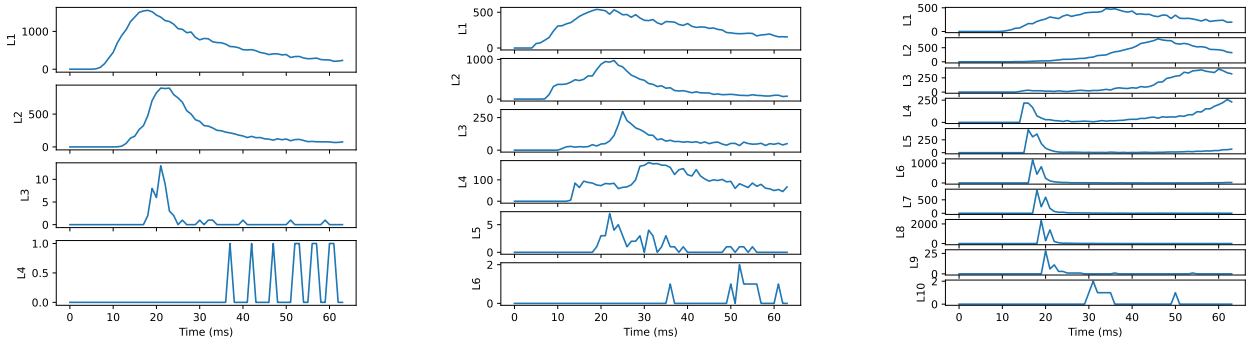
consistent behavior in which as the number of layers increases, the accuracy improves: VGG-style ANN of moderate deepness ($\sim$ 10 layers) achieved accuracy >98 % in preliminary tests. SNN exhibit a different trend: while at lower complexities they outperform ANN, showing the potential of spiking neurons in storing information, as the network deepness increases their performance peaks (at EMAC clearly lower w.r.t. the ANN counterparts), and then drops to unsatisfactory levels of accuracy. A deeper insight of this phenomenon can be obtained by looking at the internal dynamics of the networks. Fig. 7 shows the temporal trend of the number of spikes emitted by each layer of different models with different number of layers. Except for the number of layers, both the models shares the same temporal coding architecture, latency and hyperparameters. *49_TC5B* in Fig. 7a represents a case of well trained SNN: in the two convolutional levels, the layer takes a certain amount of time to collect spikes from its predecessor, after that it starts to emit spikes. The number of emitted spikes increases, reaches a maximum, and then slowly decreases, as less significant neurons fire. As it can be expected, the response of subsequent layers is slightly translated forward in time, as each layer needs to accumulate a certain amount of spikes by the layer before. Adding more layers causes the accuracy to drop due to poor training. Case *25_TC11B* in Fig. 7c is a 11 layers architecture: in fact, higher layers start to spike in advance, well before most of the information from previous layers is collected. Spike activity in layer 4 rises and reaches its peak in response of a very few input spike from the previous layer. A possible factor triggering this behavior is a too low number of time steps for the deeper networks. With same weight initialization, the time needed by the spikes to propagate among subsequent layers in deeper networks become more large, possibly larger than the available time (which is itself a hyperparameter). Backpropagation then starts only for a subset of the input, and several neurons do not even enter the training. More investigations are needed to unlock successful training of deeper networks.

4. Hardware testing

The proposed metric present also the possibility to be tuned on specific hardware platform, to achieve also absolute energy estimation. To test this possibility, a series of benchmark models were implemented on the BrainChip Akida AKD1000 [48] device, the first generation of digital neuromorphic accelerator by BrainChip. Three convolutional models of increasing complexity, differing in the number of convolutional blocks, and in the number of filters in each Conv2D layers, were trained on the EuroSAT dataset (see Fig. 8). The Akida system is digital, and its predefined way to operate consists in first training standard ANN and then convert them into SNN to be run on hardware. The ANN are trained, then quantized to match the bit depth available on hardware, and retrained to recover accuracy lost in the quantization process. Finally they are converted to spiking models. Details about the specific spiking model adopted are currently not available in the official documentation: nevertheless, integrate-and-fire behavior with ROC decoding are confirmed in [49].

4.1 Hardware performances

The ANN models reached a test accuracy value $\sim$80 % (Table 2). After the conversion to spikes, a drop in accuracy between 1.3 % (*CNN-16-32)* and 5.6 % (*CNN-16-16*) was observed, a good result considering the drastic drop in bit depth (only 4 bits are used on hardware to represent the network parameters). The AKD1000 reported an extremely low energy consumption between 0.63 and 1.38 mJ per frame during inference. The reported value for the floor (idle) power consumption was 911.49 mW, to be compared with a peak power consumption (worst case across the models) of 978.00 mW, meaning that despite the extremely low increment in power during inference, more than 93 % of the total power consumption is due to



(a) 49_TC5B – effective training. (b) 46_TC7B – suboptimal training. (c) 25_TC11B – failed training.

Fig. 7. Scaling to deeper networks with spiking CNN (ROC, IFL neurons). BNTT is applied on convolutional layers (all except the last two). As the number of layers grows, subsequent layers fail to collect most of the information of lower layers, and start to fire basing only on a few presynaptic spikes.

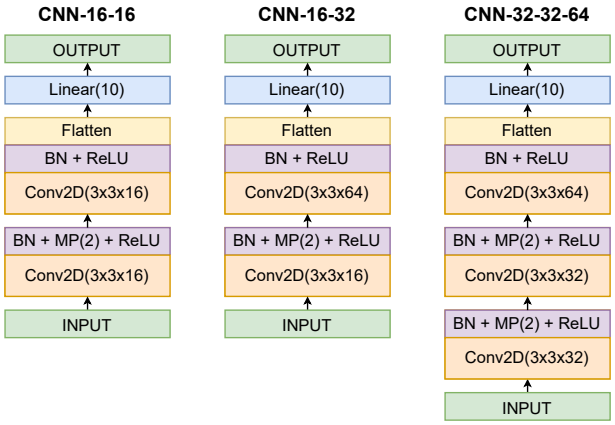


Fig. 8. Models trained for the Akida device. ANN models, prior to the conversion in SNN, are shown.

the device just being idle.

4.2 Energy model identification

The possibility of the dimensionless EMAC metric to be tuned to estimate also absolute energy consumption for specific target platform is here verified for the AKD1000, following the procedure:

1. The 3 models implemented in the AKD1000 chip are reproduced in Norse, and their EMAC is computed;
2. Two of them are used to perform a least squares estimation considering e_{syn} and e_{upd} in Eq. [2] as free parameters to fit the EMAC value to the actual energy consumption estimated by the hardware;
3. The estimated energy model is used to compute the energy consumption of the third network, which is compared with the value measured on the Akida platform to assess the achieved accuracy.

The more precisely the networks are reproduced in Norse, the higher the expected accuracy in the estimation. Since some aspects of the conversion from ANN are undocumented, some arbitrary assumptions are taken in the Norse implementation. However, such uncertainties are at least partially coped with by proper tuning of the parameters. IFL neurons which spike once at most, with 64 time steps and ROC has been selected. To reproduce a realistic network activity, the Norse models have been retrained from scratch with SG, for the weights values trained in the processor pipeline cannot be directly applied to the models. Results are shown in Fig. 9. Cases *CNN-16-16* and *CNN-16-32* have been considered in the identification of the numerical energy model then used to estimate the energy for *CNN-32-32-64*, achieving a $+5.8\,\%$ relative error: the scaling of the energy consumption due to the increase of the network size is successfully described by the now dimensional [mJ] EMAC metric. As can be expected, by using the minimum number of data points (2 for a 2 DoF model) the residuals of the least squares optimization are null and the estimated energy perfectly matches the measures at those points. The functionality of the model with a minimum data set highlights the generalization capabilities of EMAC: given more data, an even better accuracy in the estimation can be expected.

5. Conclusions

In this work, an investigation of the potential benefits of SNN for onboard Artificial Intelligence applications in space was carried out. A land use classification task, based on the EuroSAT RGB dataset and representative of a possible use in actual space missions, was selected as case

Table 2. Summary of the network performances for the Keras quantized models.

	CNN-16-16	**CNN-16-32**	**CNN-32-32-64**
Test accuracy (Keras)	79.33 %	79.41 %	85.56 %
Test accuracy (Quantized)	74.19 %	78.41 %	80.85 %
Test accuracy (Akida)	73.93 %	78.11 %	81.22 %
Floor power [mW]	911.49	911.49	911.49
Inference power (min–max) [mW]	933.00–947.00	933.00–949.59	956.00–978.00
Inference power (avg. ± sd) [mW]	944.58 ± 3.86	953.00 ± 4.42	974.83 ± 4.06
Inference energy [mJ/frame]	0.63	0.82	1.38
Avg framerate [fps]	1506.7	1154.8	707.55

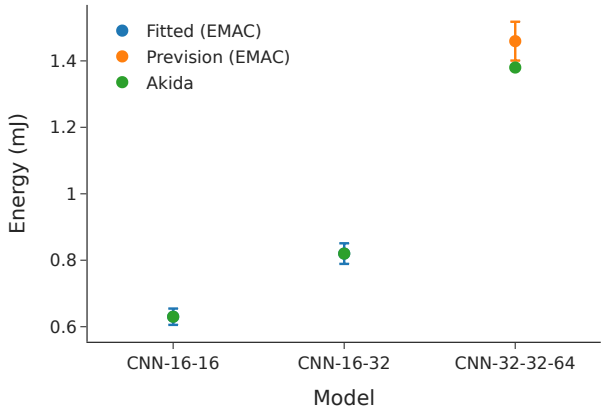


Fig. 9. Identification of the Akida energy model.

study. SNN models based on both temporal and rate coding, and their ANN counterparts, were compared in terms of accuracy and complexity in a hardware-agnostic way. A new metrics, Equivalent MAC operations (EMAC), was developed to relatively compare computational load. Differently to other popular metrics, EMAC is suitable to assess the impact of different neuron models, to weight separately contributions given by synaptic operations w.r.t. neuron updates, and to compare SNN with their ANN counterparts. By default EMAC gives a dimensionless, relative estimation, and it should be considered a proxy of energy consumption. Moreover, internal parameters can be tuned to match the features of specific hardware, if known, achieving also absolute estimation. A preliminary successful demonstration is given for the BrainChip Akida AKD1000 neuromorphic processor. Benchmark SNN models, both latency and rate based, exhibited a minimal loss in accuracy, compared with their ANN counterparts, with significantly lower (from −50 % to −80 %) EMAC per inference, making SNN of extreme interest for applications limited in power and energy typical of the space environment, especially considering that an even greater improvement (with respect to standard ANN running on traditional hardware) in energy consumption can be expected with SNN when implemented on actual neuromorphic devices. A research effort is still needed, especially in the search of new architectures and training methods capable to fully exploit SNN peculiarities.

Acknowledgements

This work was founded by the European Space Agency (contract number: 4000135881/21/NL/GLC/my) in the framework of the Ariadna research program. The EuroSAT dataset is publicly available at [30].

References

[1] G. Giuffrida *et al.*, "CloudScout: A Deep Neural Network for On-Board Cloud Detection on Hyperspectral Images," *Remote Sensing*, vol. 12, no. 14, 2020, ISSN: 2072-4292. DOI: 10.3390/rs12142205. [Online]. Available: https://www.mdpi.com/2072-4292/12/14/2205.

[2] M. Bechini, G. Gu, P. Lunghi, and M. Lavagna, "Robust spacecraft relative pose estimation via cnn-aided line segments detection in monocular images," *Acta Astronautica*, vol. 215, pp. 20–43, 2024. DOI: 10.1016/j.actaastro.2023.11.049.

[3] M. Bechini, M. Lavagna, and P. Lunghi, "Dataset generation and validation for spacecraft pose estimation via monocular images processing," *Acta Astronautica*, vol. 204, pp. 358–369, 2023, ISSN: 0094-5765. DOI: 10.1016/j.actaastro.2023.01.012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094576523000127 (visited on 01/13/2023).

[4] S. Silvestrini, M. Piccinin, G. Zanotti, A. Brandonisio, P. Lunghi, and M. Lavagna, "Implicit Extended Kalman Filter for Optical Terrain Relative Navigation Using Delayed Measurements," *Aerospace*, vol. 9, no. 9, 2022, ISSN: 2226-4310. DOI: 10.3390/aerospace9090503.

[5] S. Silvestrini *et al.*, "Optical Navigation for Lunar Landing based on Convolutional Neural Network Crater Detector," *Aerospace Science and Technology*, vol. 123, p. 107 503, 2022, ISSN: 1270-9638. DOI: 10.1016/j.ast.2022.107503.

[6] G. Meoni *et al.*, "The ops-sat case: A data-centric competition for onboard satellite image classification," *Astrodynamics*, pp. 1–22, 2024.

[7] G. Revach, N. Shlezinger, X. Ni, A. L. Escoriza, R. J. G. van Sloun, and Y. C. Eldar, "KalmanNet: Neural Network Aided Kalman Filtering for Partially Known Dynamics," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1532–1547, 2022, ISSN: 1941-0476. DOI: 10.1109/TSP.2022.3158588.

[8] A. Niitsoo, T. Edelhäußer, E. Eberlein, N. Hadaschik, and C. Mutschler, "A Deep Learning Approach to Position Estimation from Channel Impulse Responses," *Sensors*, vol. 19, no. 5, p. 1064, 5 2019, ISSN: 1424-8220. DOI: 10.3390/s19051064. [Online]. Available: https://www.mdpi.com/1424-8220/19/5/1064 (visited on 06/28/2023).

[9] G. Furano *et al.*, "Towards the Use of Artificial Intelligence on the Edge in Space Systems: Challenges and Opportunities," *IEEE Aerospace and Electronic Systems Magazine*, vol. 35, no. 12, pp. 44–56, 2020, ISSN: 1557-959X. DOI: 10.1109/MAES.2020.3008468.

[10] B. Han, A. Ankit, A. Sengupta, and K. Roy, "Cross-Layer Design Exploration for Energy-Quality Tradeoffs in Spiking and Non-Spiking Deep Artificial Neural Networks," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 4, pp. 613–623, 2018, ISSN: 2332-7766. DOI: 10.1109/TMSCS.2017.2737625.

[11] M. Bouvier *et al.*, "Spiking Neural Networks Hardware Implementations and Challenges: A Survey," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 15, no. 2, 22:1–22:35, 2019, ISSN: 1550-4832. DOI: 10.1145/3304103. [Online]. Available: https://dl.acm.org/doi/10.1145/3304103 (visited on 06/29/2023).

[12] S. R. Kheradpisheh and T. Masquelier, "Temporal Backpropagation for Spiking Neural Networks with One Spike per Neuron," *International Journal of Neural Systems*, vol. 30, no. 06, p. 2 050 027, 2020. DOI: 10.1142/S0129065720500276.

[13] A. R. Voelker, D. Rasmussen, and C. Eliasmith, *A Spike in Performance: Training Hybrid-Spiking Neural Networks with Quantized Activation Functions*, http://arxiv.org/abs/2002.03553, ArXiv:2002.03553, 2021, preprint.

[14] C. Stöckl and W. Maass, *Recognizing Images with at most one Spike per Neuron*, https://arxiv.org/abs/2001.01682, ArXiv:2001.01682, 2020.

[15] S. R. Kheradpisheh, M. Mirsadeghi, and T. Masquelier, "BS4NN: Binarized Spiking Neural Networks with Temporal Coding and Learning," *Neural Processing Letters*, vol. 54, no. 2, pp. 1255–1273, 2022, ISSN: 1573-773X. DOI: 10.1007/s11063-021-10680-x. [Online]. Available: https://doi.org/10.1007/s11063-021-10680-x.

[16] E. Hunsberger and C. Eliasmith, *Training Spiking Deep Networks for Neuromorphic Hardware*, http://arxiv.org/abs/1611.05141, ArXiv:1611.05141, 2016.

[17] S. Park, S. Kim, B. Na, and S. Yoon, *T2FSNN: Deep Spiking Neural Networks with Time-to-first-spike Coding*, http://arxiv.org/abs/2003.11741, ArXiv:2003.11741, 2020.

[18] J. Göltz *et al.*, *Fast and energy-efficient neuromorphic deep learning with first-spike times*, http://arxiv.org/abs/1912.11443, Arxiv:1912.11443, 2021.

[19] N. Caporale and Y. Dan, "Spike Timing–Dependent Plasticity: A Hebbian Learning Rule," *Annual Review of Neuroscience*, vol. 31, no. 1, pp. 25–46, 2008. DOI: 10.1146/annurev.neuro.31.060407.125639. pmid: 18275283. [Online]. Available: https://doi.org/10.1146/annurev.neuro.31.060407.125639 (visited on 06/29/2023).

[20] P. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in Computational Neuroscience*, vol. 9, 2015, ISSN: 1662-5188. DOI: 10.3389/fncom.2015.00099. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fncom.2015.00099.

[21] A. Tavanaei, T. Masquelier, and A. S. Maida, "Acquisition of visual features through probabilistic spike-timing-dependent plasticity," in *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016, pp. 307–314. DOI: 10.1109/IJCNN.2016.7727213.

[22] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, S. J. Thorpe, and T. Masquelier, "Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks," *Pattern Recognition*, vol. 94, pp. 87–95, 2019, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2019.05.015.

[23] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, and T. Masquelier, "SpykeTorch: Efficient Simulation of Convolutional Spiking Neural Networks With at Most One Spike per Neuron," *Frontiers in Neuroscience*, vol. 13, p. 625, 2019, ISSN: 1662-453X. DOI: 10.3389/fnins.2019.00625.

[24] B. Illing, J. R. Ventura, G. Bellec, and W. Gerstner, "Local plasticity rules can learn deep representations using self-supervised contrastive predictions," in *Advances in Neural Information Processing Systems*, 2021, pp. 1–15.

[25] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–8. DOI: 10.1109/IJCNN.2015.7280696.

[26] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019, ISSN: 1558-0792. DOI: 10.1109/MSP.2019.2931595.

[27] A. S. Kucik and G. Meoni, "Investigating spiking neural networks for energy-efficient on-board ai applications. a case study in land cover and land use classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2020–2030.

[28] S. Davidson and S. B. Furber, "Comparison of artificial and spiking neural networks on digital hardware," *Frontiers in Neuroscience*, vol. 15, p. 651 141, 2021.

[29] I. M. Comsa, K. Potempa, L. Versari, T. Fischbacher, A. Gesmundo, and J. Alakuijala, "Temporal Coding in Spiking Neural Networks with Alpha Synaptic Function," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 8529–8533. DOI: 10.1109/ICASSP40776.2020.9053856.

[30] P. Helber, B. Bischke, A. Dengel, and D. Borth, *EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification*, http://arxiv.org/abs/1709.00029, ArXiv:1709.00029, 2019.

[31] F. Zenke and S. Ganguli, "SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks," *Neural Computation*, vol. 30, no. 6, pp. 1514–1541, 2018, ISSN: 0899-7667. DOI: 10.1162/neco_a_01086.

[32] F. Zenke and T. P. Vogels, "The Remarkable Robustness of Surrogate Gradient Learning for Instilling Complex Function in Spiking Neural Networks," *Neural Computation*, vol. 33, no. 4, pp. 899–925, 2021, ISSN: 0899-7667. DOI: 10.1162/neco_a_01367.

[33] J. Rossbroich, J. Gygax, and F. Zenke, "Fluctuation-driven initialization for spiking neural network training," *Neuromorphic Computing and Engineering*, 2022. DOI: 10.1088/2634-4386/ac97bb; 10.48550/arXiv.2206.10226.

[34] S. B. Shrestha and G. Orchard, *SLAYER: Spike Layer Error Reassignment in Time*, http://arxiv.org/abs/1810.08646, ArXiv:1810.08646, 2018.

[35] C. Pehle and J. E. Pedersen, *Norse - A deep learning library for spiking neural networks*, https://github.com/norse/norse, Accessed: 2023-06-27, 2021.

[36] E. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1063–1070, 2004, ISSN: 1941-0093. DOI: 10.1109/TNN.2004.832719.

[37] T.-J. Yang, Y.-H. Chen, J. Emer, and V. Sze, "A method to estimate the energy consumption of deep neural networks," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, 2017, pp. 1916–1920. DOI: 10.1109/ACSSC.2017.8335698.

[38] 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. DOI: 10.1109/ISSCC.2014.6757323.

[39] B. Degnan, B. Marr, and J. Hasler, "Assessing Trends in Performance per Watt for Signal Processing Applications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24,

no. 1, pp. 58–66, 2016, ISSN: 1557-9999. DOI: 10.1109/TVLSI.2015.2392942.

[40] M. Davies *et al.*, "Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 911–934, 2021, ISSN: 1558-2256. DOI: 10.1109/JPROC.2021.3067593.

[41] W. Guo, M. E. Fouda, A. M. Eltawil, and K. N. Salama, "Neural Coding in Spiking Neural Networks: A Comparative Study for Robust Neuromorphic Systems," *Frontiers in Neuroscience*, vol. 15, p. 212, 2021, ISSN: 1662-453X. DOI: 10.3389/fnins.2021.638474.

[42] Y. Kim and P. Panda, "Revisiting Batch Normalization for Training Low-Latency Deep Spiking Neural Networks From Scratch," *Frontiers in Neuroscience*, vol. 15, 2021, ISSN: 1662-453X. DOI: 10.3389/fnins.2021.773954. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2021.773954 (visited on 12/22/2022).

[43] E. Lemaire, L. Cordone, A. Castagnetti, P.-E. Novac, J. Courtois, and B. Miramond, "An Analytical Estimation of Spiking Neural Networks Energy Efficiency," in *Neural Information Processing*, 2023, pp. 574–587. DOI: 10.1007/978-3-031-30105-6_48.

[44] M. Dampfhoffer, T. Mesquida, A. Valentian, and L. Anghel, "Are SNNs Really More Energy-Efficient Than ANNs? an In-Depth Hardware-Aware Study," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 7, no. 3, pp. 731–741, Jun. 2023, ISSN: 2471-285X. DOI: 10.1109/TETCI.2022.3214509.

[45] J. Schemmel, S. Billaudelle, P. Dauer, and J. Weis, *Accelerated Analog Neuromorphic Computing*, http://arxiv.org/abs/2003.11996, ArXiv:2003.11996, 2020.

[46] W. Banerjee, R. D. Nikam, and H. Hwang, "Prospect and challenges of analog switching for neuromorphic hardware," *Applied Physics Letters*, vol. 120, no. 6, p. 060 501, 2022, ISSN: 0003-6951. DOI: 10.1063/5.0073528. [Online]. Available: https://doi.org/10.1063/5.0073528 (visited on 06/23/2023).

[47] G. Indiveri *et al.*, "Neuromorphic Silicon Neuron Circuits," *Frontiers in Neuroscience*, vol. 5, 2011, ISSN: 1662-453X. DOI: 10.3389/fnins.2011.00073. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2011.00073 (visited on 06/23/2023).

[48] *Brainchip akida white paper*, https://brainchip.com/wp-content/uploads/2023/11/Bacteria-in-Blood-research-paper_Final.pdf, Accessed: 2024-01-19.

[49] B. M. Posey, *What Is the Akida Event Domain Neural Processor?* https://brainchip.com/what-is-the-akida-event-domain-neural-processor-2/, Accessed: 2024-03-13, BrainChip, 2022.