Received 5 December 2023, accepted 19 December 2023, date of publication 25 December 2023, date of current version 29 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3346429

RESEARCH ARTICLE

On-the-Fly Learning With Mixed-Mode Spiking Neural Network and Passive Memristive Array: Application to Neuromorphic Cameras

PIERRE LEWDEN[1], ADRIEN F. VINCENT[2], JEAN TOMAS[2], AND SYLVAIN SAÏGHI[1,2]
[1]CNRS@CREATE, Singapore 138602
[2]Univ. Bordeaux, CNRS, Bordeaux INP, IMS, UMR 5218, F-33400 Talence, France

Corresponding author: Sylvain Saïghi (sylvain.saighi@u-bordeaux.fr)

This work was supported in part by the European Union's Horizon 2020 Research Innovation Programme (Ultra-Low Power Event-based Camera (ULPEC) Project) under Grant 732642; in part by the French National Research Agency (ANR); in part by the "Chaires Intelligence Artificielle (IA)" Program (Green Artificial Intelligence (GrAI) Project) under Grant ANR-19-CHIA-0003; in part by the "Programmes et Équipements Prioritaires de Recherche (PEPR) IA" Program (Emergences Project); in part by the Programme DesCartes; and in part by the National Research Foundation, Prime Minister's Office, Singapore, under its Campus for Research Excellence and Technological Enterprise (CREATE) Programme.

ABSTRACT The massive deployment of the Internet of Things (IoT) combined with the need to reduce its impact on global energy consumption calls for the design of intelligent sensors. These sensors must be able to process information in situ to reduce the amount of data to be transferred, and to perform computing at low energy cost. Therefore, the design of intelligent sensors requires a compromise between performance and energy consumption. Spiking neural networks (SNNs) are good candidates for achieving the goal of an efficient intelligent sensor, as they use event-based computation. Interest for hardware implementation of SNNs has bloomed since the first experimental observation of memristors in 2008. In this paper, we study, by simulation means, the impact of the main technological parameters of Ferroelectric Tunnel Junctions (FTJs) synapses and of analog Leaky Integrate-and-Fire (LIF) neurons on the system learning capabilities. This allows us to determine which parameters are critical for the design of such systems and to suggest mitigation solutions as well as guidelines on how to build a SNN-based smart vision sensor in the context of unsupervised or reward-modulated learning. In particular, we show that splitting up the passive crossbar array of memristors could help dealing with the detrimental effect of the input voltage offset of the postsynaptic neurons.

INDEX TERMS Edge computing, memristors, passive crossbar, reward-modulated learning, spiking neural networks, unsupervised learning.

I. INTRODUCTION

Artificial Neural Networks (ANNs) and Deep Learning (DL) [1] have revolutionized machine learning in many fields like healthcare, automotive, speech and visual recognition. To attempt to achieve human performances and surpass them on specific tasks [2], [3], ANNs have evolved into a much more complex structure since their first implementation by Rosenblatt in 1958 [4]. The past 60 years have seen the birth of Deep and Convolutional Neural Networks (DNNs, CNNs) used in computer vision, which have added many layers of computation to solve complex tasks. However, these neural networks were mostly implemented on Von-Neumman architectures, which are not built for the highly parallel tasks that ANNs aim to achieve. To solve this problem, other computational architectures such as Graphical Processor Unit (GPUs) were diverted from their original purposes [5], before processor units or hardware accelerators were built, such as Tensor Processing Units (TPU) [6] or Vision Processing Units (VPU) [7] dedicated to the optimization in terms of power and speed of some of the mathematical operations used by ANNs.

The associate editor coordinating the review of this manuscript and approving it for publication was Hadi Tabatabaee Malazi.

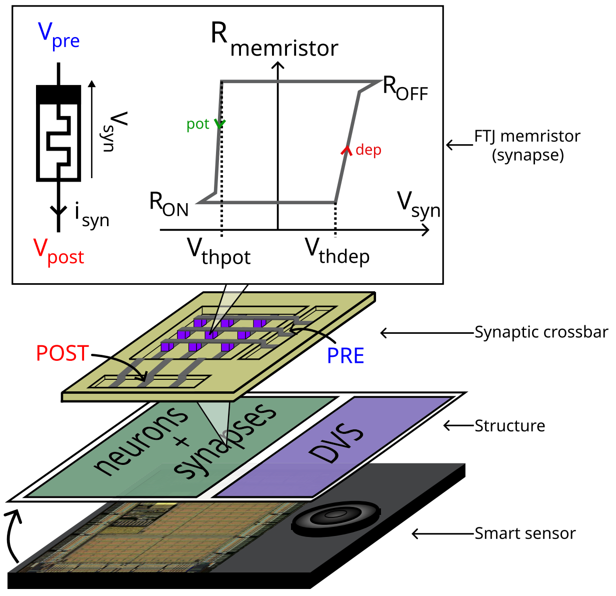


FIGURE 1. Illustration and structure of theoretical visual smart sensor for edge computing. It combines a DVS camera for the sensing element and a mixed-mode SNN made of analog neurons and a crossbar array of memristive synapses.

Spiking Neural Networks (SNNs) [8], another type of ANN closer to the brain [9], [10], are expected to be much more energy-efficient, like their biological counterpart, making them good candidates for embedded systems or smart sensors dedicated to enabling edge computing, where the computation is carried out as close as possible to the acquisition of the data. Their computational framework, based on the use of discrete electrical signals (events), is a very different approach to that of the widely used CNNs using continuous signals, thus leading to a number of different challenges when targeting neuromorphic hardware implementation. This energy-efficient idea raised the question of either convert already-working ANNs to SNNs [11] or adapting the learning rules used in those other neural networks. However, despite the growing interest in SNNs, their actual applications are limited by the sparsity of event-based sensors that can natively communicate with them, for example the Dynamic Vision Sensor (DVS) [12].

The interest in hardware spiking neural networks has increased in recent years, leading to the construction of dedicated chips as ‘‘neuromorphic processors’’, such as TrueNorth [13], [14], Loihi [15] or other ‘‘neural emulators’’ [16] that can be used to deploy or emulate SNNs. However, they are mostly developed as versatile chips that need to be configured or tuned to work for a specific application. The question of a purpose-built neuromorphic smart sensor with on-line learning for dedicated applications made to optimize power efficiency and implementation cost is still an open topic, with many unanswered questions. These questions include which neural network architecture to use, the type of learning rules, and the technology used to implement the neurons and synapses. While the origin of neuromorphic systems was the use of analog component behaviors for event-based computing, today's systems are either mixed (analog neurons and digital synaptic communications), like BrainScale [17] and Neurogrid [18], or entirely digital, like SpiNNaker [19]. However, the emergence of another type of device in the 2010s, called the memristor, reshuffled the deck.

If neurons (the ‘‘computation units’’) are a critical part of neuromorphic systems, synapses (the ‘‘memory points’’) are actually one of the main remaining technological bottlenecks as they are much more numerous than neurons (e.g., by a factor 1,000× to 10,000× in the human brain [20], [21], [22]). Besides, emulating learning-capable synapses, for which some form of synaptic plasticity is required, often severely degrades integration density especially for silicon-based synapses [23]. Memristors combine two of the main properties of biological synapses: they are a memory point and are intrinsically plastic. As well as their ability to emulate synapses, we can add their nanometric dimensions, allowing high integration in dedicated chips. We will not debate here the merits of any particular memristor technology [24], as we believe that each technology has its place depending on the intended application, just as we now have several memory technologies in the same digital system. Still, in this work we have chosen ferroelectric tunnel junction memristors (FTJs) because they offer interesting characteristics when interfaced with an integrated circuit. We selected a threshold voltage of the order of a volt, a resistance in the order of a megohm (therefore a current of the order of a microampere), and a ratio between Ron (lowest resistance state) and Roff (high resistance state) that can be in the order of a hundred (Roff/Ron≈100) [25], [26]. An M×N crossbar array of nanoscale memristors can emulate a dense synaptic connection layer between M presynaptic and N postsynaptic neurons allowing high integrability (see Fig. 1). One sometimes adds a selection transistor in series with each memristor to make a 1T1M structure, which reduces both sneak path currents and the current needed to drive the array during programming events [27], [28]. However, such a solution requires a significant amount of extra control signals (for the transistors) degrading the integrability. That is why for this work we have chosen to consider a passive crossbar array of memristors, without any selection device, in order to simplify manufacturing and to allow for higher integration density. This choice also shifts the complexity of the system realization to the microelectronics part.

In previous work, we defined by simulation a set of hardware-friendly mixed mode unsupervised and reward-modulated learning rules that enable learning of real-word data without accelerating the task timescale [29], [30] of a SNN, using a passive memristive array and analog neurons. The present work aims to study the influence of different parameters of a silicon neural network with ferroelectric memristive synapses on the learning capabilities of such a network in the context of the co-integration of an event-based visual sensor with this neural network. Section II presents the context of the study, i.e., the system block diagram, the learning rules and the database. Section III focuses on the memristors, detailing their model parameters and then the influence of their initialization values and the transition asymmetry between Ron and Roff values on learning capabilities. Section IV studies the impact of a few tunable parameters and the variability of some of the microelectronic subcircuits on the network performance.

II. CONTEXT

A. GENERAL ARCHITECTURE

The type of low power smart visual system studied in this work is composed of two parts:

- The "sensing" part for the data acquisition is achieved through a dynamic vision sensor that asynchronously outputs events according to the light variation that each one of its pixels detects. The use of a dynamic vision sensor is justified by its ability to natively emit spikes preventing the need for encoding layers or any algorithmic overhead.
- As our aim is to build a first demonstrator, and given the difficulty of co-integrating FTJ memristors on chips, edge computing is achieved through a fully connected single-layer spiking neural network fed by the output of the dynamic vision sensor to detect some learned patterns as soon as possible.

Furthermore, we consider the following scenario where the smart sensor learns once deployed ("on-the-fly"), meaning that the smart part of the sensor will carry out the learning in a local manner with data acquired in real time.

The general architecture of the smart sensor is shown in Fig. 2. The edge computing part is composed of an all-to-all analog spiking neural network with 1156 input neurons (34 × 34) and 100 output Leaky Integrate-and-Fire (LIF) neurons. The presynaptic neurons are predefined waveform generators. When an event is received for the DVS, they apply a waveform dedicated to inference and, when a training phase occurs, they apply different waveforms depending on the targeted change of weight (see Appendix C for further details). On top of similar waveform generator capabilities, the postsynaptic neurons (detailed in Section IV-A) also include the electronic circuits required for implementing and driving the LIF behavior.

This analog SNN is controlled by a Digital Control Block (DCB), making it a mixed-mode architecture. The DCB, a logical block introduced in previous work [29], [30], collects the data (events) coming from the acquisition block (sensor data) and the output events of the postsynaptic analog neurons. Using this collected data, the DCB achieves multiple sub-functions to enable the sensor to learn and recognize data. Some of those sub-functions serve:

- to determine which postsynaptic neuron fired using an event arbiter that ensures that only one postsynaptic neuron is chosen if more than one fired during the same timestamp (see Appendix A);

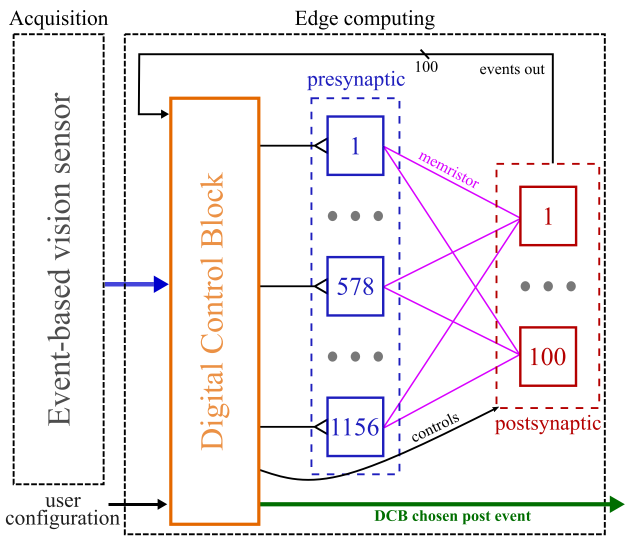


FIGURE 2. General architecture of the smart sensor composed of a Digital Control Block, an input layer of 1156 neurons, an output layer of 100 Leaky Integrate-and-Fire (LIF) neurons and an all-to-all memristive passive array.

- to trigger a predefined presynaptic waveform used for inference according to the events coming from the acquisition part;
- to trigger predefined presynaptic and postsynaptic waveforms, according to the learning rules it implements (see Appendix C);
- to disable a postsynaptic neuron if it did not pass the labeling phase or should not be used.

The results presented in this article were obtained by means of an in-house Python simulator. Appendix A describes the main computation principles used in this software tool.

B. THE DATASET

As the aim of this work is to determine through simulation all the parameters and guidelines to follow before implementing the actual sensor, a dataset derived from a dynamic vision sensor is needed. We use the N-MNIST dataset generated by filming a MNIST handwritten digits dataset with a DVS performing 3 successive saccades [31]. This dataset is composed of 60,000 training samples and 10,000 test samples for 10 classes (digits from 0 to 9). While each sample contains two polarities of events (ON/OFF) depending on the perceived light variation (increase/decrease), our study limits itself to the ON events to reduce the number of events needed for computation and the number of inputs needed, as the inputs of our spiking neural network only use one type of event. Furthermore, as the task we aim to achieve is to only recognize the digits, we only feed into our network the first 100 ms, corresponding to the first saccade of the sensor used to generate the dataset. Indeed, while each sample corresponds to a single digit, some studies have seen the different saccades as different classes (or sub-patterns) [32], as the pixels composing the digits are not in the same general location due to the movement. This can become a challenge for very low power architecture that does not go through multiple layers with convolution and pooling to remove the location dependence such as ours, which only uses one all-to-all layer. Thus, each saccade is seen as a different class, leading to either a reduction in the number of available output neurons per class, or an increase in the overall number of outputs. For our task, this data trimming (ON events only, first 100 ms) of the original dataset leads us to reduce drastically the number of events fed into the SNN. Removing the OFF events, especially, should lead to less activity used and thus less power used when computing the sensor information. An example of such a N-MNIST sample (with our data trimming) is shown in Fig. 3.

Finally, as the aim in this work is to recognize the digit as fast as possible, when a postsynaptic neuron fires, the next sample is given even if there is still data in the on-going sample that was not consumed by the spiking neural network. Doing so also prevents another kind of similar problem that leads to using only the 100 ms. As the perceived digit is moving (relatively to the frame, as it is the camera that moves for this dataset), the digit appears at multiple and slightly different locations over the 100 ms, as can be seen in Fig. 3 where the digit appears at different locations every 10 ms, which, in a very simple system, would most likely lead to more sub-classes. Thus, our system will learn to recognize the digit at the beginning of the camera displacement.

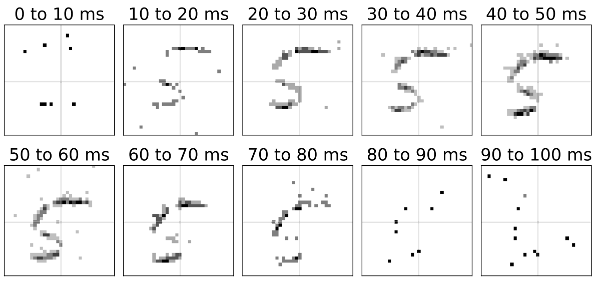


FIGURE 3. **N-MNIST sample with our data trimming. For each image, the events have been accumulated over a 10 ms time-window. The first and last time-windows correspond respectively to the beginning and end of the mechanical movements (camera displacement) leading to a decreased number of events.**

C. LEARNING RULES

To enable the sensor to learn by itself in a local manner with actual data while respecting the idea of low power consumption, one must define learning rules that do not require a lot of computation to be implemented. Unsupervised learning using only local learning rules (rules only using the activity of the neuron connected by a synapse) is a prime candidate, as no information about the actual weight is needed, meaning that in our system, no circuit dedicated to acquiring the values of the memristors' conductances is needed.

Because we intend to use a short inference window (called T_{LTP}, around a few microseconds) to stimulate the postsynaptic neurons while keeping power consumption low, we cannot effectively learn the N-MNIST dataset with the standard STDP (reinforce the events falling into the inference window and depreciate all the others), as only a sparse number of input events fall into this time window [29] for the samples used, which come from a slow mechanical movement (relatively to the inference time window). These few coincident events are not sufficient to discriminate or recognize a digit. To solve this problem, in our previous work we introduced a type of unsupervised learning ruled called iPjD, where, when a postsynaptic event occurs, all the connected synapses that transmitted at least i presynaptic events are potentiated (iP) and those that fired fewer than j events are depreciated (jD). This learning rule uses one memory cell per presynaptic input stored in the DCB, called N_{fire}, which registers for each input how many presynaptic events were emitted per input since the last postsynaptic event occurred. This removes the inference time window mismatched with the event-based data issued by the DVS.

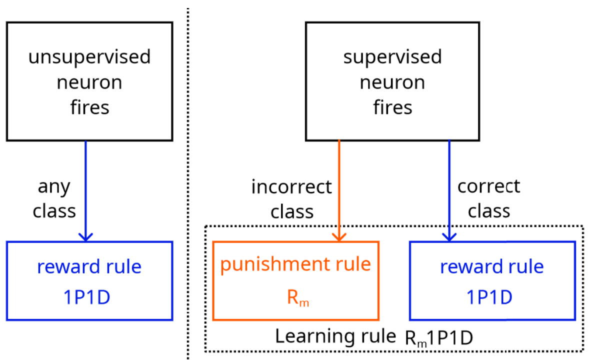
These rules allow the system to learn while keeping the learning algorithm overhead to a minimum thanks to their simplicity, as they only require knowledge of how many events were emitted recently by the presynaptic inputs. Building SNNs using unsupervised or reward modulated learning rules is encouraged by results in similar works [33], [34], [35] using the sign of the pre and postsynaptic time delays instead of the number of presynaptic events.

In this work, the unsupervised rule 1P1D, where only a 1-bit memory is needed per input, is used, as adding more bits with the trimmed dataset only decreased performances [30]. This rule potentiates synapses connected to inputs that fired at least once and depreciates those that had never fired since the last postsynaptic event. In addition to the unsupervised 1P1D rule, we also use a set of reward modulated rules introduced beforehand, which change the applied learning rule depending upon the sample class that a postsynaptic neuron fires on, as illustrated in Fig. 4. These rules can also be called "weak supervision", as they do not need to compute any kind of error or need to acquire the value of the synapses, unlike standard back-propagation algorithms.

In this work, we consider two additional learning rules given in Fig. 4 to allow some supervision in the system. The $R_\gamma 1P1D$ learning rule applies a fraction of the unsupervised reward rule $1P1D$ opposite if a neuron fires on the wrong class, while $R_\varnothing 1P1D$ consist of doing nothing (no weight change) for a wrong class.

D. LABELING AND PERFORMANCE

To define the performance of the smart sensor after a training phase, the postsynaptic neurons must have a defined label (to which class they are sensitive) to compute the performances in a test phase where all the postsynaptic neurons that successfully learned something are put in competition with



Reward rule	1P1D	
$N_{fire} \geq 1$	potentiate	
$N_{fire} < 1$	depreciate	
punishment rule	R_m	
m	γ	$\varnothing$
$N_{fire} \geq 1$	dep	null
$N_{fire} < 1$	null	null

FIGURE 4. **Unsupervised and weakly supervised learning rules. According to the value of the N_{fire} memory cell, when a postsynaptic neuron fires, the sign of the synaptic weight change is decided. The weak supervision rules presented here correspond to a fraction of the reward learning rule opposite (R_γ) or simply a skip of the weight modification ($R_\varnothing$).**

each other. In the case of unsupervised learning, the question of the postsynaptic neuron labels is unavoidable. Instead of presenting the training dataset a second time without modifying the weights to determine which neuron is sensitive to which class, we use a simple and economical heuristic method using the latest outputs of the networks in the training phase to prevent any unnecessary power consumption. Using the outputs obtained in the training phase:

- If a postsynaptic neuron fired fewer than 50 times (among the 60,000 samples), it is disabled and will not be used in the test phase.
- If in the last 50 accounted events of a postsynaptic neuron, one class fired the most and represents more than $1/N_{class}$, it is taken as a label. If more than one class fired the most and the same number of times, the neuron is disabled and will not be used in the test phase.

In the case of weakly supervised learning, this heuristic can cause a small number of output neurons to get labels that are not the ones that were targeted, a phenomenon we called mislabeling. If a mislabeling occurs on a postsynaptic neuron, we still force the label to be the one we wanted to learn. This forced labeling did not show any significant difference in terms of performance when we compared the two versions in the reference situation. This forced labeling only takes place when one wants to supervise (impose) the output label. The results in this work for the rules $R_\gamma 1P1D$ and $R_\varnothing 1P1D$ are given using this method.

E. PERFORMANCE OF THE REFERENCE SITUATION

The purpose of the sensor we aim to create is to be a low-cost, event-based smart sensor that can execute on-line learning to recognize digits in our study case. As it is aimed towards low-cost implementation and power efficiency, the architecture and the learning rules it implements were kept as simple as possible. This choice involves acknowledging a certain trade-off between:

1) The cost of the sensor in terms of power and implementation cost.
2) The recognition rate we can obtain.

Thus, while we expect the sensor to be able to recognize the digits, we cannot expect the results obtained to match the performances (recognition rate) of state-of-the-art CNN architectures implemented on GPUs.

While the details of the reference parameters and their choices will be given in Section III for the ferroelectric devices and in Section IV for the neurons, in Table 1 we give the reference results we obtain in this paper with our architecture. The results are averaged for five sets of random (uniform law) initial synaptic weights for the three learning rules of interest.

TABLE 1. **Results of our reference situation for the three learning rules for 5 sets of random initial synaptic weights.**

Learning Rule	min	avg	max
1P1D	64.65 %	65.60 %	66.47 %
$R_\varnothing$1P1D	73.00 %	73.57 %	74.30 %
R_γ1P1D	74.13 %	74.78 %	75.16 %

From these reference results, we first note that the architecture is indeed able to recognize the digits, as it is substantially above a random guess when discriminating $N_{classes} = 10$. Furthermore, as expected, the weak supervision greatly outperforms the unsupervised learning, while in the case of $R_\varnothing$1P1D it should consume less power because it does not go through a weight change when a postsynaptic neuron fires on the wrong class. The rule R_γ1P1D should have a power consumption in between 1P1D (unsupervised) and $R_\varnothing$1P1D as it only changes a subset of the weights if a neuron fire on the wrong class to depreciate them. One might argue that these results are not as high as those from other works [32] with very few layers, however:

- We showed in our previous work [30] that adjusting the parameters (learning rate) and increasing the number of outputs allows us to increase the recognition rate, meaning that the reference situation does not represent the maximum possible recognition rates. But doing this will have consequences in terms of implementation cost and power consumption.
- We chose to have lower performances for a higher level of integration and to consume less power. This is the trade-off between performance and power consumption that we mentioned previously.

III. FTJ MEMRISTIVE ARRAYS AS SYNAPTIC MATRICES

Ferroelectric Tunnel Junction (FTJ) memristors can be made using different ferroelectric materials [36], [37] and will, depending on their size and the interface materials, show different weight ranges and have different modification parameters (modification threshold, minimal pulse width to change them, etc.). The conductance change of these devices can be achieved through the repetition of fixed amplitude impulsions of a short time width of a few hundred nanoseconds [37], or even under the nanosecond [38]. The high synaptic conductance state that will consume the most current when inferring data can be around a few microsiemens while having physical dimensions of a few hundred nanometers [26], [37], making these devices ideal for fast, low power and highly integrable applications.

For our application, we consider the progressive weight change that can be achieved by this repetition of short programming pulses [25], [37] that will be applied at every training phase, making the system and the synapses able to change (learn) progressively without being restrained by a specific learning time window thanks to our learning rules. As we do not wish want to restrict ourselves to a specific technology of ferroelectric memristor (stack of materials used), the learning rule we use is a synthetic version that only captures the general behavior of the synapses. One can expect such a system to be resilient to synaptic variability as suggested in the literature [39].

In this section, we study two important issues for the passive memristive crossbar array. The first one concerns the impact of the asymmetrical behavior between the increase and the decrease in the memristor resistance on the learning performance that can be intrinsic to the technology (see Fig. 1) or forced by adapting the waveforms used. The second one focuses on the impact of different initialization weights on the capabilities of learning.

A. SYNAPSES MODEL

We assume that the weight modification is achieved through the application of repetitive fixed impulsions, as it has been observed in some ferroelectric memristor technologies if those impulsions are below a negative device writing threshold V_{thpot} to increase the conductance, i.e., the synaptic weight (assuming $V_{thpot} \leq 0$), or above the positive device writing threshold V_{thdep} to decrease the conductance (assuming $V_{thdep} \geq 0$). These two thresholds V_{thpot} and V_{thdep} (see Fig. 1) define the amplitude range $\left[V_{thpot} \leq 0; V_{thdep} \geq 0\right]$ of the possible amplitudes that can be applied without changing the synaptic weights (conductances), which is especially important for the inference phase but also for the weight modification phase of a passive memristive array. In our architecture, to achieve the weight modification of a device in a crossbar, the presynaptic and postsynaptic neurons are required to simultaneously trigger specific waveforms, as shown in Fig. 5.

This method allows us to modify only the memristor of interest by splitting the voltage between the presynaptic and the postsynaptic connections. Using this one third/two thirds method, the memristor that will be modified will see the full

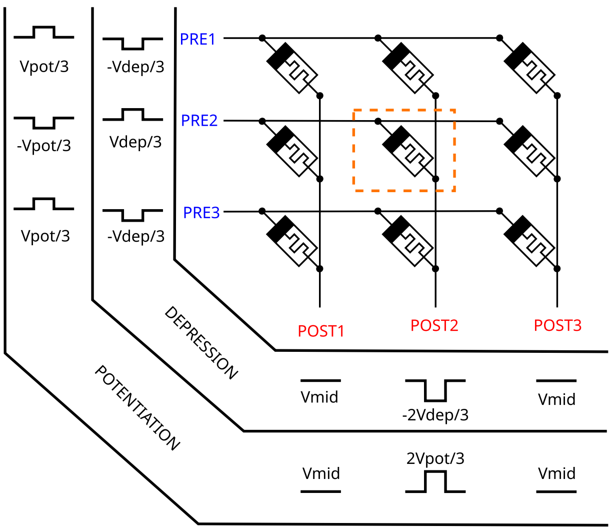


FIGURE 5. Example of the voltages used to perform inference or modify the synaptic weights of one memristor (dashed orange area) in a 3 × 3 passive array. The PRE and POST terminals indicate the crossbar lines and columns towards the presynaptic and postsynaptic neurons.

programming voltage V_{pot} or V_{dep} to increase or decrease the weight, while the other memristors in the array will only see one third of the programming voltage. This ensures that the other synaptic weights are not modified, while providing more margin in the maximum impulsion amplitudes that can be used to change a synaptic weight compared to a half-voltage programming scheme. This method makes three assumptions:

1) All the presynaptic and postsynaptic connections of the crossbar are maintained to a potential at all times with a resting value of V_{mid}.
2) The voltages V_{pot} and V_{dep} are higher than their respective writing thresholds (V_{thpot} and V_{thdep}) for the memristive device.
3) $\pm V_{pot}/3$ and $\pm V_{dep}/3$ are lower than the writing thresholds of the memristors in the passive array.

These three conditions must be fulfilled to be able to change the synaptic weight of interest. Finally, using this method, not all the afferent weights can be increased or decreased at the same time for a postsynaptic neuron, leading to a minimum of two writing phases. Furthermore, maintaining a voltage on every presynaptic and postsynaptic line (see Fig. 5) is also necessary to reduce as much as possible any sneak path current (uncontrolled/unwanted current flowing inside the crossbar).

To describe in our simulation the modification triggered when a postsynaptic fires while not restricting us to a very specific memristive implementation, we use a simple self-limiting model for the conductance change ΔG given in Equation (1), which qualitatively fits the behavior of several memristive technologies where G_o is the current synaptic conductance, G_{min} is the minimum possible conductance and G_{max} is the maximum possible conductance.

A_{pot} and A_{dep} are the potentiating and depression learning rates which may depend on the waveform duration or amplitude applied on the memristor ($V_{syn} = V_{pre}$-V_{post}).

$$\Delta G = \begin{cases} +A_{pot} \times (G_{max} - G_o) & \text{for } V_{syn} \leq V_{thpot} \\ -A_{dep} \times (G_o - G_{min}) & \text{for } V_{syn} \geq V_{thdep} \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

B. THE LEARNING RATES ASYMMETRY

The conductance range we use in our simulations ([$G_{min} = 10\,\text{nS}$; $G_{max} = 1\,\mu\text{S}$] reported in Appendix B Table 5) was chosen as the reference situation. It is based on the possible values that can be achieved by ferroelectric memristors, as these are fast and highly resistive devices, thus limiting the overall energy consumption. Still, lower or higher synaptic values could be used assuming that some of the neuron parameters are adapted, as will be discussed in Section IV. In our simulations, unless specified otherwise, conductance values are randomly initialized from a uniform distribution between G_{min} and G_{max}. The learning rates A_{pot} and A_{dep} are a more open question as their value depends on the memristor technology and the waveform (amplitude, width) used to change the conductance. However in our application case, there are two elements that can give a general idea of their effective range:

1) IN CASE OF VERY LOW LEARNING RATES

Very low learning rates (A_{pot} and $A_{dep} \leq 3.33\,\%$ for example) will likely lead to the need for more train samples or the use of multiple epochs of a dataset (number of presentations of the full train dataset to the network) for the system to fully specialize and finish learning. Furthermore, such a low learning rate also assumes that the memristors implemented in the architecture can be modified with a high level of precision, without any means of acquisition to tune them in our case.

2) IN CASE OF VERY HIGH LEARNING RATES

Very high learning rates ($\geq 50\,\%$ for example) might not be achievable due to the nature of the memristor passive crossbar. Depending on the technology, a high learning rate might be linked to a higher writing amplitude or longer impulsion time width. Indeed, in a passive memristive crossbar, the possible writing amplitude is limited by the threshold writing voltages of the memristive devices, as the other devices that are not being updated will see a residual amplitude that needs to be lower than the writing threshold.

An example of the synaptic weight evolution with our synthetic rule is shown in Fig. 6.

To see the impact of these learning rates for the learning rules 1P1D (unsupervised), $R_{\varnothing}$1P1D and R_{γ}1P1D, we run the simulations for 5 sets of random (uniform law) initial synaptic weights. Fig. 7 shows the average performance obtained for the three learning rules for different values of A_{pot} and A_{dep}, where the red square area locates the reference situation, presented in Section II-E, and the green square

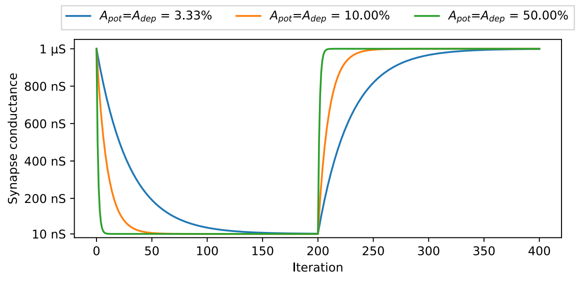


FIGURE 6. Weight evolution using different pairs of learning rates (A_{pot}, A_{dep}) of a single synapse when applying 200 consecutive potentiations followed by 200 depressions.

area shows where the maximum averaged learning rate was obtained.

For the three learning rules, we can see that an asymmetry between A_{pot} and A_{dep} in favor of A_{pot} gives better performances for the chosen task (recognizing N-MNIST numbers), as the maximum averaged performances for every learning rule is when A_{dep} is at $3.33\,\%$ (minimum value tested) and a A_{pot} at $13.33\,\%$ for 1P1D, $16.66\,\%$ for $R_{\varnothing}$1P1D and $10\,\%$ for R_{γ}1P1D.

As some FTJ memristor technologies can naturally present an asymmetry in favor of A_{pot} to increase the conductance [25], the fact that our system handles an asymmetry in favor of A_{pot} better than one in favor of A_{dep} is encouraging for an actual implementation using those devices.

The rule R_{γ}1P1D that generally yields better results for low values of A_{dep} shows a sudden drop in performances, down to $0\,\%$ (fail-stop condition) when A_{dep} is too high. This fail-stop condition stops the simulated architecture if, for 50 successive samples, the architecture did not fire any output events. This can be explained by the behavior of the punishment rule $R\gamma$1P1D, which decreases all weights that contributed to charging a postsynaptic neuron learning a wrong class. As multiple pixels are common between the different classes, if those common pixels are decreased too strongly (after multiple depressions due to the punishment rule), the neuron will not be able to charge enough until it is unable to fire any longer.

The rule $R_{\varnothing}$1P1D yields globally the best performances, with a strongly degraded performance when the learning rate asymmetry is in favor of A_{dep} (weights are strongly decreased). Furthermore, it is the rule that theoretically consumes less current, as when a neuron fires on the wrong class, there is no programming phase.

The rule 1P1D works best with a small asymmetry in favor of A_{pot} and low learning rates.

The simulations carried out for this paper used identical learning rates for all the synapses in the system. However, in a real implementation, it is unlikely that every memristive device will demonstrate the same learning rates. To see the impact on the system's performances, we use randomly chosen learning rates A_{pot} and A_{dep} between $6.66\,\%$ and $13.33\,\%$ (around the reference situation) using a uniform rule to see the impact of such behavior. The range of these learning rates is depicted by the purple area in Fig. 7. To see the effect of such a situation, we run 5 sets of simulations for different initial synaptic weights and randomly chosen learning rates for the 3 learning rules of interest. The results obtained are given in Table 2.

TABLE 2. Simulation results for the 3 learning rules obtained with 5 sets of random initial synaptic weights and the variability over A_{pot} and A_{dep}.

Learning Rule	min	avg	max	Ref avg*
1P1D	63.9 %	64.97 %	66.27 %	65.50 %
$R_{\varnothing}$1P1D	73.73 %	74.08 %	74.52 %	73.57 %
R_{γ}1P1D	74.67 %	75.33 %	75.79 %	74.78 %

* Average reference situation results from Table 1.

As the results obtained with or without the learning variability are close, we can conclude that the architecture in itself is resilient to such a behavior, which is encouraging as having devices with the exact same learning rates is unlikely due to device variability. Our results are consistent with results from other works that mention the resilience of spiking neural networks to synaptic variability [39].

C. INITIALIZATION OF THE MEMRISTIVE ARRAY

When deploying the neuromorphic system for its first use or for a new task, the question of the initial synaptic weights must be addressed. The reference situation and most of the simulations presented in this work give us an estimate of the smart sensor's behavior with synaptic weights initialized in a random fashion with a uniform distribution. With a minimalist architecture such as ours that do not have a complex acquisition and tuning system, this kind of weight distribution seems difficult to achieve. A starting situation that is simpler to implement as it should not require any weight measurement unit to be added, relies on the idea of starting from a simple reference situation like a checkers pattern, where half of the weights are connected to a postsynaptic neuron close to their minimum or maximum values. This initial situation can be achieved by triggering multiple times the learning phase over a dedicated pattern that we want to learn. However, having (or setting) half of the synaptic weights at their highest conductance weight would lead to a high power consumption, as the more weight needed to be in their ON state, the more current would be needed in the inference or training phase. Thus, we considered different initial patterns where only memristors chosen on a periodic basis among the number of inputs would be set to their highest value G_{max} (yellow in Fig. 8), while all the others are set to their minimum conductance values G_{min} (black in Fig. 8).

Fig. 8 shows different initialization patterns set for every postsynaptic neuron (initial conductance/sensitivity map) and the results obtained after training the SNN using these initial patterns. The patterns used correspond to synaptic weights set to their maximum G_{max} value following a chosen period between 2 (half of the memristors set to G_{max}) and 10 (every 10th memristor set to G_{max}), while all the other weights are set to their minimum value G_{min}.

From the results we obtained with different initializing patterns, we can observe that the lower the number of synapses in their high conductance state (yellow), the more the learning rate drops. Indeed, if too many synaptic weights start in a low conductance state, some postsynaptic neurons can end up in a situation where they are unable to fire due to low synaptic currents.

We can see that having 1 device set to G_{max} every 3 memristors (33.33 % of the synapses, period 3) gives very similar results after 1 epoch compared to the reference situation (see Table 1). We can also observe that for some of the initialization patterns, some simulation ended as failures. This means that the simulation has ended up with weights that do not stimulate the postsynaptic neurons enough to fire for 50 consecutive presented samples.

To go slightly further, we also consider setting a percentage (33.33 %) of randomly chosen inputs that will have synapses at their maximum values, while the other inputs are connected to a synapse at its minimum conductance, instead of using a forced pattern. From our results given in Table 3, and we can see that one of the simulations for $R\gamma$1P1D ended in the fail-stop condition. Indeed, choosing a random distribution of the initial synaptic weights with only a small fraction of them not in a low conductance state increases the chances of starting the system with a distribution of weights where the weights set in their high conductance states are not connected to the input where the useful data will be located, leading to postsynaptic neurons being unable to charge enough to fire and learn. This is the same reason why some of the fixed initialization patterns led to failure.

TABLE 3. Simulation results for the 3 learning rules obtained with 5 sets of initial synaptic weights with a randomly chosen percentage of input set to G_{max} with the rest set to G_{min}. The patterns are the same for each postsynaptic neuron.

Learning Rule	percentage	min	avg	max
1P1D	50 %	63.59 %	65.04 %	66.70 %
	33 %	64.87 %	65.79 %	66.18 %
	25 %	63.07 %	64.84 %	65.56 %
$R_{\varnothing}$1P1D	50 %	73.48 %	73.68 %	74.06 %
	33 %	72.54 %	73.22 %	73.46 %
	25 %	73.22 %	73.51 %	74.80 %
R_{γ}1P1D	50 %	74.21 %	75.02 %	75.85 %
	33 %	fail	74.40 %*	76.13 %
	25 %	71.87 %	73.91 %	74.80 %

* Average obtained by removing the simulation that ended in a failure.

To seek to prevent this situation where synaptic weights do not permit the postsynaptic neurons to fire often enough, one can fill the presynaptic N_{fire} memory with a pattern of choice (which may be different from one postsynaptic to another), then trigger the postsynaptic neuron of interest to force it to learn a initializing pattern. This method relies on the existing learning protocols inside the DCB, and only a small logic overhead to allow the filling of N_{fire} and of an

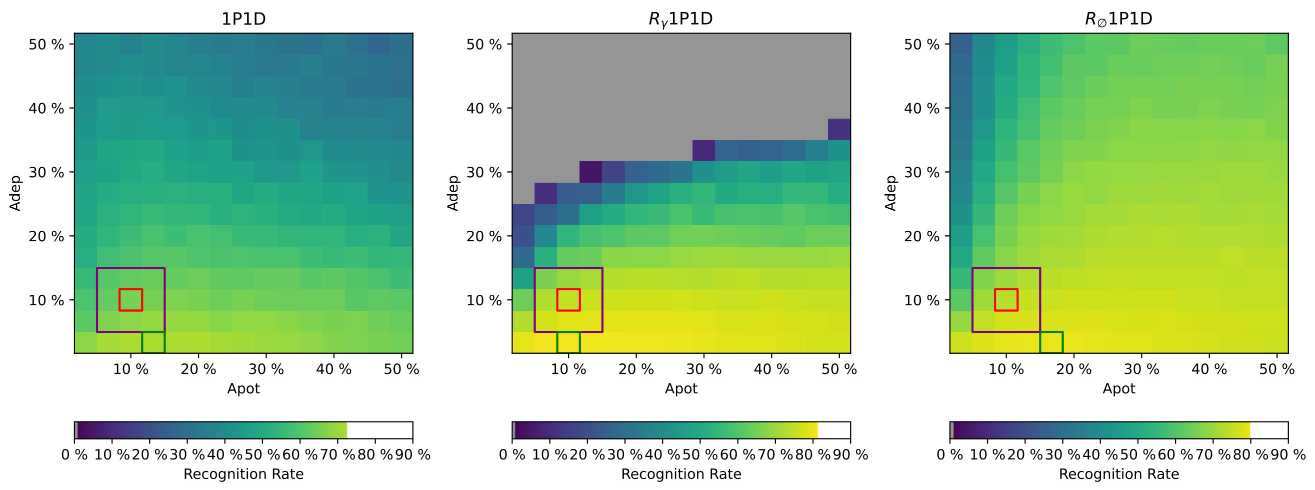


FIGURE 7. Averaged performances for different learning rate combinations obtained after 1 epoch. The red area represents the results of the reference situation. The green area represents the location of the maximum learning rate. The purple area represents the range of the uniformly distributed A_{pot} and A_{dep} for the results given in Table 2. The color bars are filled up to the (average) maximum learning rate obtained for the learning rule.

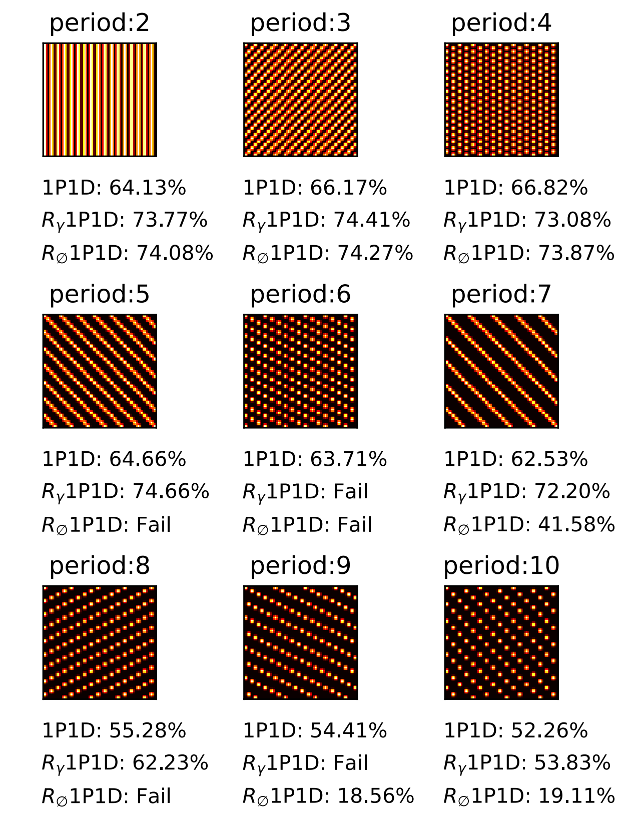


FIGURE 8. Results obtained with the network starting with different initial synaptic weight patterns. The pattern shown is the same for every postsynaptic neuron in the network. Yellow represents synapses at G_{max}, black represents synapses at G_{min}.

additional memory containing the initialization pattern would be required.

To verify the impact of this kind of initialization, we define the following test in simulation using a reduced architecture (30 postsynaptic neurons instead of 100, as we will learn only 3 classes):

1) Randomly set the weights to start the simulation.
2) Initialize each postsynaptic neuron by triggering the learning phase multiple times when the N_{fire} memories are filled according to the initialization pattern for every postsynaptic neuron.
3) Learn a subset of the N-MNIST (digits 5, 6, 9).
4) Initialize each postsynaptic neuron with the initialization pattern (same as step 2).
5) Learn a different subset of the N-MINIST (digits 0, 1, 4).
6) Initialize each postsynaptic neuron with the initialization pattern (same as step 2).
7) Learn the previous subset of the N-MINIST (digits 5, 6, 9) to ensure that we can retreive similar performances to those obtained previously.

The results we obtained with this protocol are shown in Fig. 9 where, for one set of initial synaptic weights, the conductance map of the first postsynaptic neuron is shown and the corresponding recognition rate of the reduced architecture for each learning rule is given. Using this method, the time required for an initialization will be the time to fill the N_{fire} memories plus the number of times the learning phase is triggered times the number of postsynaptic neurons if they are all initialized sequentially. In simulation we only triggered 25 programming phases to learn the initialization pattern, which is enough to make the previous sensitivity maps appear, but not enough to make them completely disappear, meaning that 750 training phases were needed for this reduced architecture (only 30 postsynaptic neurons) to initialize it.

Despite not fully erasing the previous sensitivity maps beacause not all the weights after the initialization are at

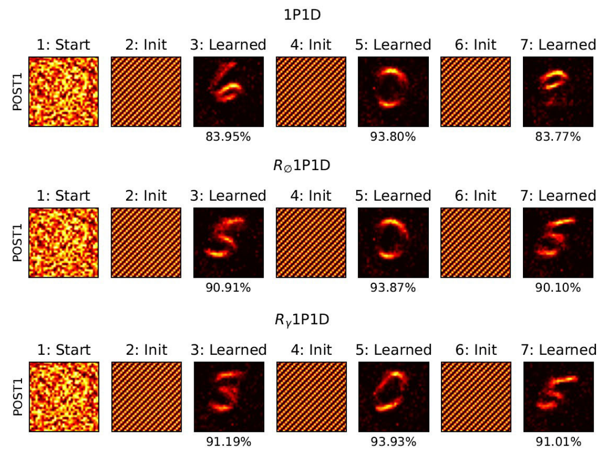


FIGURE 9. Results obtained for 1 set of initial synaptic weights with the reduced architecture using the initialization process pattern 'Period=3'. The conductance matrix of the first postsynaptic neuron is shown with the resulting Recognition Rate of the full SNN (3 classes, 30 postsynaptic neurons).

their min or max values (not visible in Fig. 9), as we do not trigger enough training phases to reach the minimum or maximum conductance states for every synapse, we do achieve similar recognition rates between steps 3 and 7, meaning that our programming protocol would allow us to change the application of a deployed sensor even if it was used for a different task. Of course, if one wants to completely forget the previous task, one can simply increase the number of training phases used to program the initialization pattern, in the knowledge that this will require more time and power.

Finally, it is worth mentioning that the initialization phase is not linked to the learning rule that will be used, as no inference is performed. Indeed, even if an inference was used to fill the N_{fire} memory, as the same pattern is used to program the neurons one by one, the neuron can only fire on a "correct" class.

IV. DESIGN CHOICES AND LEVERAGES

The chosen spiking neural network is a single all-to-all layer with 1156 inputs (34 × 34 to match the dataset shape) and 100 output neurons connected through a memristive synaptic crossbar array. While the previous section addressed the synaptic array, how to use or initialize it and what impact its behavior and characteristics would have on the overall smart sensor, this section focuses on the impact of the chosen neuron parameters on the SNN performance. The SNN has two types of analog neurons, presynaptic ones that will stimulate the synapses according to the data from the acquisition part, and postsynaptic neurons that will classify the data. Furthermore, both presynaptic and postsynaptic neurons apply dedicated waveforms when a weight needs to be modified for training or initialization (see Appendix C).

To progress from the model of the sensor to an actual implemented architecture, one must first define the actual parameters that will be used. Instead of running multiple random batches of simulations to find a working situation, in this part, we consider the main neuron parameters and their restrictions used to determine the working reference situation while looking at the impact some of those parameters can have on performances.

Table 5 in Appendix B contains the reference values of the spiking neural network parameters.

A. POSTSYNAPTIC NEURONS MODEL USING A CCII+

As the target of our simulations is a hardware implementation of the studied sensor using analog neurons and synapses, our output neuron model is defined by an electrical circuit shown in Fig. 10. The values of its parameters are constrained by physical values.

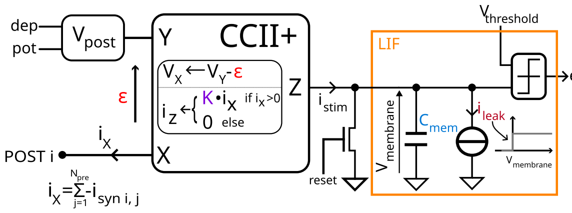

dep
pot
V_{post}
Y
CCII+
$V_X \leftarrow V_Y - \varepsilon$
$i_Z \leftarrow \{ K \cdot i_X$ if $i_X > 0$; 0 else
Z
X
ε
i_X
POST i
$i_X = \sum_{j=1}^{N_{pre}} i_{syn\ i,j}$
i_{stim}
reset
LIF
$V_{membrane}$
C_{mem}
i_{leak}
$V_{threshold}$
event

FIGURE 10. Postsynaptic neuron model. The V_{post} unit constantly applies a potential V_{mid} unless a potentiation (pot) or depression (dep) signal is received to apply dedicated weight modification waveforms. Both signals pot and dep are driven by the DCB unit. POST i represents the connected column of the synaptic array where all the memristors connected to the same postsynaptic neuron are electrically connected.

The chosen neuron model is a Leaky Integrate-and-Fire (LIF) neuron, a variation of an Integrate-and-Fire neuron which is the least complex spiking neuron model to implement [40]. Therefore, this LIF neuron model is ideal for low-cost and efficient computation. The "Leaky" part is assured by a current source (transistor-based circuit) instead of a classical resistor, allowing the use of adjustable, very low-discharge current without using too much surface, because the lower the discharge current is, the more surface a resistor would take up in an integrated circuit. The "Integrate" part is assured by an analog capacitor, while the "Fire" part is assured by a comparator circuit that emits an output event if the membrane potential $V_{membrane}$ reaches $V_{threshold}$. The postsynaptic event signal is collected by the DCB in charge of controlling the presynaptic and postsynaptic neurons if a synaptic weight update must be executed.

The input synaptic current used to charge the membrane of the LIF neuron is collected through a second-generation positive current conveyor (CCII+). The use of a CCII+, proposed by G. Lecerf [41], enables the collection and copying of the sum of synaptic currents connected to the same postsynaptic neuron (Kirchhoff's law), while constantly applying a postsynaptic voltage or waveform (V_{post}). Thanks to this property, we can effectively apply different presynaptic and postsynaptic waveforms to either infer data to stimulate the postsynaptic neurons or modify the synapses' weights without requiring any switching circuits connected to the synaptic array. In our case, only positive synaptic currents (I_X) are copied to the output Z of the CCII+ ($I_Z \leftarrow K \cdot I_X$) to prevent any discharge of the neuron membrane. The voltage applied on the postsynaptic terminal of the synaptic array is always fixed at V_{mid} (resting potential), except when a postsynaptic neuron is to have its connected synaptic weights updated according to the protocol described previously (see Fig. 5).

The postsynaptic voltage copy offset ϵ of the CCII+ represents the variability of components of the input differential pair. It can critically impact the behavior of the overall neural network, as mentioned in our previous work [29]. This offset, which may differ according to the postsynaptic neuron, causes the generation of currents in the connected synapses, leading to some adverse effects that will be mentioned later. Unless specified otherwise, it is fixed at zero for all the postsynaptic neurons in our simulations.

The postsynaptic neurons also possess a reset circuit to either put the membrane at its resting potential 0 V or to disable a postsynaptic neuron if needed. Furthermore, the postsynaptic neurons do not possess the standard refractory period usually used to prevent one postsynaptic neuron from concentrating all the network activity. Indeed, while a refractory period would need to be adjusted according to the input data dynamic to be efficient, in our previous work we introduced a refractory counter N_{refrac} that is only active during a training phase to remove this constraint linked to the synaptic stimulation dynamic. When a postsynaptic neuron fires in a training phase, it follows the weight update protocol before being disabled for N_{refrac} events issued from other postsynaptic neurons. This method effectively prevents a postsynaptic neuron from firing consecutively for at least N_{refrac}. In our simulations, we fixed this N_{refrac} counter at 10, which is a tenth of the number of outputs.

1) MEMBRANE CAPACITANCE

The membrane capacitance C_{mem} is the component that implements the "Integrate" part of the analog LIF neuron described in this architecture. The bigger it is, the more space it will need in the analog neuron. If it is too small, the time constant of the postsynaptic neuron will become smaller until it is too small for the neuron to charge enough if the input data dynamic is too slow (incoming events are spread through a large time window). Considering all of this, we choose to run our simulation with a C_{mem} of 1 pF [42].

We showed in previous work that our architecture is resilient to membrane capacitance variability with a slightly different SNN configuration and use [29]. Table 4 shows, for the current architecture, the average, minimum and maximum results obtained in this work for 5 simulations with different starting weights and postsynaptic neuron capacitances. Our results show that a strong variability of these membrane capacitances (uniform law between ±20 % of C_{mem}) does not have a big impact on the recognition rate.

TABLE 4. Results for the three learning rules over 5 simulations with different initial synaptic weights and membrane capacitances. The membrane capacitance are chosen following a uniform law between 0.8 pF and 1.2 pF.

Learning Rule	min	avg	max	Ref avg*
1P1D	62.43 %	64.80 %	66.84 %	65.60 %
$R_{\varnothing}$1P1D	71.99 %	72.73 %	73.00 %	73.57 %
R_{γ}1P1D	73.74 %	74.83 %	75.67 %	74.78 %

* Average reference situation results from Table 1.

2) MEMBRANE THRESHOLD AND DISCHARGE CURRENT

The membrane threshold $V_{threshold}$ implements the "Fire" part of the analog LIF neuron. It is another leverage on the neurons' behavior. As we assume that the implemented sensor is working using a standard 5 V or even 3.3 V power source, we choose a typical value of 1 V that can potentially be adjusted by an external polarizing circuit if needed.

To implement the "Leaky" part of the analog LIF neuron we use a current source pumping out a $i_{discharge}$ current from the membrane capacitance if its voltage is above 0 V. In our simulation, we use a discharge current $i_{discharge}$ of 100 pA. It corresponds to a discharge speed of $i_{discharge}/C_{mem}$ = 0.1 V ms^{-1} or 1 V (the membrane threshold) per 10 ms, which is a tenth of the duration of the trimmed sample we use in which a digit can be visible (see Fig 3). Along with the discharge speed, we fixed the discharge current at a value that would consume the current incoming by a single OFF state memristive device ("worst stimulation case").

B. CHOOSING THE SYNAPTIC STIMULATION STRENGTH

In our neuron model, the CCII+ that connects the memristive crossbar array to the postsynaptic neurons copies the synaptic current with a factor K. This parameter, used to adjust the synaptic stimulation of the postsynaptic membrane, can be estimated according to the range of the synaptic weights (conductances), the type of data we wish to recognize and the neuron parameters such as the membrane capacitance C_{mem}, the neuron threshold voltage $V_{threshold}$, the absolute amplitude of the inference voltage Δv_{stim} (see Appendix B and C) and the neuron discharge current $i_{discharge}$. If we consider a single synapse at its maximum value G_{max}, the current it generates if stimulated once during an inference is $i_{syn} = G_{max} \times \Delta v_{stim} = 1\,\mu$A for an inference time window $T_{LTP} = 10\,\mu$s. The increase in the membrane potential it would represent without a current copy adjustment ($K = 1$) is $\Delta V_{membrane} = T_{LTP} \times (K \times i_{syn} - i_{discharge})/C_{mem} = 9.999$ V, which is almost 10 times the threshold voltage. To prevent this situation there are a few possible solutions:

- Increase the membrane capacitance. However, this choice would lead to a higher surface use in the circuit per postsynaptic neuron.
- Decrease the stimulation voltage. It could be a good solution to also reduce the power consumption. However, by lowering it too much (around a few millivolts), the circulating currents through the crossbar would be

much smaller and it could become problematic due to intrinsic noise of the devices.
- Decrease K, the current copy factor of the CCII+, to reduce the current that will be injected into the neuron membrane. This is mathematically equivalent to decreasing the voltage stimulation (Δv_{stim}) without reducing the current collected by the CCII+.

In our simulation we decided to leverage the use of the K by reducing it. It is fixed to $K = 1/100$ for the reference situation, meaning that a synapse at its maximum value would increase the membrane by 99 mV. A synapse at the average value of 505 nS would increase the membrane by 49.5 mV, while one at its OFF state (10 nS $\rightarrow$100 pA injected in the membrane) would not alone increase the membrane potential due to the value of the discharge current (100 pA) that was given before. Looking at these values, one might argue that a stimulation in the maximum (average) state would lead to only 10 (20) simultaneous events to trigger a postsynaptic. However, when choosing the value of this K (or Δv_{stim}) parameter, one must take into account the dynamic of the type of data we want to recognize, because the events do not come at the same time and the LIF neuron discharges over time. This input data dynamic must be taken into account to adjust how much the postsynaptic neurons are stimulated.

The strength of the stimulation of the LIF neurons inside the spiking neural network of the smart sensor is adjusted according to the synaptic weights range [$G_{min} = 10$ nS; $G_{max} = 1$ µS], the stimulation amplitude Δv_{stim} and the factor K. When deploying the sensor, multiple scenarios impacting the stimulation strength of the implemented LIF neurons could arise. Among them are:

- A shifted operating range of the synaptic weights. Depending on the technology they could show higher or lower conductance synaptic weights.
- One might try to reduce Δv_{stim} to reduce the power consumption of the inference phase as long as it is strong enough to charge the postsynaptic neurons.

To see the impact of these possibles changes, we simulate the architecture for multiple possible values of K for the three learning rules of interest, as it is equivalent to either changing the synaptic weights or changing the Δv_{stim}. Fig. 11 shows the simulation results for K taken between $K/2$ and $K \times 2$, which is equivalent to simulating with Δv_{stim} between $\Delta v_{stim}/2$ and $\Delta v_{stim} \times 2$ or between the synaptic range divided by two and multiplied by two.

According to our results shown in Fig. 11, the recognition rate starts to drop the smaller the value of K is. This is consistent with the problem mentioned earlier. A smaller stimulation strength means that less current will be injected in the membrane, and the discharge will become too strong for more synaptic stimulation, preventing the membrane from reaching the threshold. Conversely, the stronger the stimulation of the membrane (K at higher values), the faster the neuron will charge and trigger an output event. This is also a problem as it leads to having fewer inputs (pixels) used to

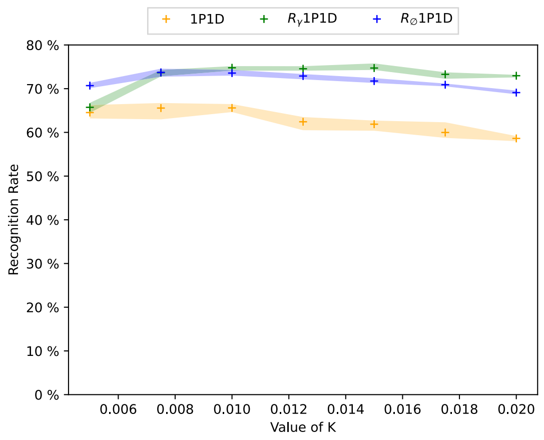


FIGURE 11. **Impact of the averaged performance on the smart sensor depending on the strength of the stimulation by changing the current scaling factor K. Each point (+) represents the average results obtained. The tinted area around each point represents the minimum and maximum values obtained.**

discriminate the data (digits in our case), until too few inputs are used (not enough pixels used to discriminate a digit in our case).

Considering the limitation on both sides, we can see that the SNN is still quite tolerant to different stimulation strengths, as there is no apparent narrow optimum of the recognition rate. While one might want to adjust the stimulation strength to get maximum performances, the system will still be able to work efficiently if fine tuning is not possible.

The unsupervised learning rule seems to be able to handle lower stimulation strength, while showing a stronger degradation of performance if the stimulation strength is increased. Thus, one could, with this learning rule, reasonably decrease the inference absolute amplitude Δv_{stim} to reduce power consumption without losing too much performance.

The weakly supervised rule $R_\varnothing$1P1D shows a similar behavior to the unsupervised learning rule, but with a much higher recognition rate. Still, its performance when decreasing the stimulation strength drops faster than the unsupervised learning rule.

The rule R_γ1P1D can handle higher stimulation strengths better than the other two rules. However, if the stimulation strength drops, its performance drops abruptly towards the performance of the unsupervised rule. This behavior could be explained by the nature of the rule, which depreciates all the contributing weights as a punishment if a neuron fires on the wrong class. Because the data (digits) share common inputs for the dataset we use, this leads to a reduced synaptic stimulation of those shared contributing weights each time the neuron is "wrong", which, in addition to a general reduced stimulation strength seems to be too much of a punishment for the value tested in the simulation.

C. RESILIENCE TO THE INPUT DATA DYNAMIC

The iPjD learning rules and their weakly supervised variation were made to be less dependent on the input dynamic than the classical STDP. This means that the sensor should still be able to recognize the sensor despite events occurring faster. To see how our architecture behaves for different input dynamics, we run simulations for different accelerated factors of the trimmed data we used, by dividing the samples' timing by a speed factor $acc = \{1, 10, 100\}$ for the training and the test phases. This change of input dynamic should impact the performances because if the stimulation events are occurring faster, keeping the parameters of the architecture as they are means that fewer events would be needed for the neurons to trigger, leading to fewer inputs used to try to recognize the data and less current would then be consumed to trigger an answer from the SNN. To influence or compensate for the postsynaptic neurons' charge dynamic of the SNN, a few parameters could be altered: $C_{membrane}$, K (or Δv_{stim}) and $i_{discharge}$, which impacts how quickly the postsynaptic neurons reacts. However, while some parameters should be easier to adjust on a deployed sensor through some polarization and tuning circuits, the analog membrane capacitance is supposedly a fixed physical capacitor in our architecture. Due to this, we only consider altering the $i_{discharge}$, which describes how fast the neurons discharge. The simulation results (5 simulations for 5 initial synaptic weight distributions) for the three acceleration factors mentioned and different discharge currents (from the reference 100 pA to 1 nA) for the three learning rules are given in Fig. 12.

For the unsupervised 1P1D learning rule, if we consider the reference configuration of the SNN ($i_{discharge}$ =100 pA), we can see that having events occurring at a higher speed causes a decrease in performance, but it is still able to work and recognize almost 60 % of the digits on average, and increasing the discharge current allows the architecture to regain some performance. Furthermore, if the data is accelerated by 10, multiplying by 10 the discharge speed allows the architecture to restore the reference situation performance. When accelerating by a hundred, increasing the discharge by ten does increase the performance compared to the original discharge speed, but the reference situation results are not achieved. The two other weakly supervised learning rules can handle much better the accelerated dataset at the reference situation, as the performance only drops slightly compared to the unsupervised rule, while R_γ1P1D handles it better than the other rules. The leakage current $i_{discharge}$ is a useful lever if one has to deal with a faster input dynamic, and using one of the proposed weak supervision learning rules can also prevent the loss of as much performance as the unsupervised learning when dealing with a faster input dynamic.

Concerning the original speed of the dataset ($acc = 1$), we can see that when the data has been accelerated, the recognition rate drops sharply the higher $i_{discharge}$ is. And conversely to the behavior with a faster input rate, the unsupervised rule is more resilient to an increase in the discharge current as it drops slower than R_γ1P1D and $R_\varnothing$1P1D. The reason it drops at the original speed for the three rules is because a higher $i_{discharge}$ means a higher stimulation strength would be needed for sparse events to be able to trigger a postsynaptic event, as mentioned before.

Thus, according to our results, when dealing with relatively slow data inputs, one should be sure to have a small enough $i_{discharge}$, but if the input data is coming at a much faster rate, increasing it can actually optimize the performance.

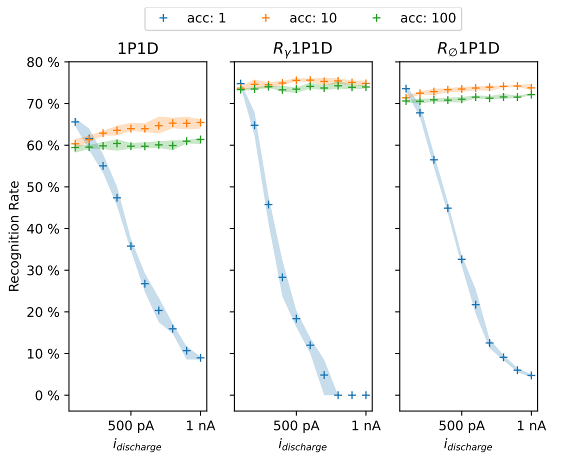


FIGURE 12. **Averaged performance when the input dynamic is altered by an acceleration factor for different values of leakage current $i_{discharge}$. Each point (+) represents the average results obtained. The tinted area around each point represents the minimum and maximum values obtained.**

D. SPLITTING THE SYNAPTIC CROSSBAR

Dealing with a passive crossbar while offering an interest in terms of integration comes with some constraints and limitations. Among those limitations:

- Because all lines (presynaptics) or columns (postsynaptics) are connected all together to the corresponding inputs, the neuron circuits in charge of driving them must be able to drive the equivalent parallel resistance during inference, but also to provide the current needed to modify the weights in a training phase. Due to this, having too many memristors connected on a same terminal can require too much current to be achievable.
- Because the connection between two successive memristors on the same line or column will show a certain resistance r_x (that should be as small as possible to limit its impact), the more memristors are connected together, the more the memristors will see a voltage drop along the line or column connection.
- The more memristors are connected to the same postsynaptic neuron, the greater the impact of the voltage copy offset ϵ will be for the network.

This voltage copy offset ϵ is the representation of the variability of the postsynaptic neuron integration that can, if uncontrolled, cause the architecture to have defective postsynaptic neurons. Its impact is strongly linked to the connected synapses, as it will cause the generation of synaptic current in the architecture. Depending on its sign, there are two possible different impacts:

- If it is in the negative range, it will cause a synaptic current to be continuously injected into the membrane, and could result in spurious output events if it is above the $i_{discharge}$ current.
- If it is in the positive range, it will cause the generation of a current in the crossbar that will shadow mask some of the synaptic activity if it is too great. Furthermore, because the CCII+ can only copy positive currents (pumped from the X terminal according to the normalization, see Fig. 10) the "negative" current generated will only shadow mask some synaptic activity and not discharge the membrane.

To alleviate those difficulties linked to how many memristors are connected in parallel, changing the architecture slightly can reduce their impact. In our case, as it is the number of inputs that is the largest and will thus causes more challenges, the solution we consider, as mentioned in the literature [43], is to split the main crossbar or synaptic array into smaller ones to separate the presynaptic inputs. Using multiple CCII+, we can collect and copy to the corresponding postsynaptic neurons the synaptic currents issued by the smaller crossbars. Reducing the number of connected memristors at the same output neurons with this method can drastically reduce the detrimental effect linked to this high parallelism, as they originally connect to all the synaptic inputs (1156 in our case). Fig. 13 shows an altered postsynaptic neuron using one CCII+ per smaller crossbar to combine the different currents. However, one must keep in mind that this solution brings an additional cost in area and power, as more electronic circuits are needed.

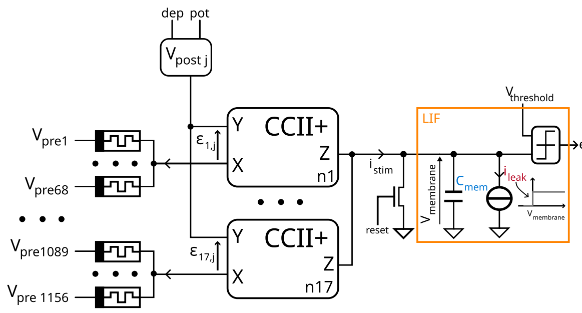


FIGURE 13. Altered structure for a postsynaptic neuron with $N_{xbar} = 17$. The partial output synaptic currents can be combined by connecting the output terminal of the CCII+ thanks to Kirchhoff's law.

As the first two limitations mentioned (current requirement of a synaptic neuron and voltage drop) are highly linked to the fabrication process and technology of the ferroelectric memristors, we only look at the impact of splitting the crossbar array on the detrimental voltage copy offset. We only consider positive voltage copy values, as:

- if it is negative, it would constantly generate current towards the neuron membrane (as long as this current is higher than the discharge);
- if it is positive, it would generate a current that absorbs the synaptic activity as it would be in opposition;
- we assume that a dedicated adjustment circuit could be added to reduce it to a maximum and set it up in the right range.

Fig. 14 illustrates the simulation results we obtained when the ϵ is randomly chosen following a uniform law between 0 mV and different upper bound values ϵ_+ (1 mV, 2 mV, 3 mV and 4 mV) for every CCII+ in the architecture when it is using 1 crossbar of 1156 × 100, 2 of 578 × 100, 4 of 289 × 100, and 17 of 68 × 100. Fig. 14 shows on the top panels the recognition rate of the architecture for these different situations, while the bottom panel gives the number of useful neurons after training (with a label).

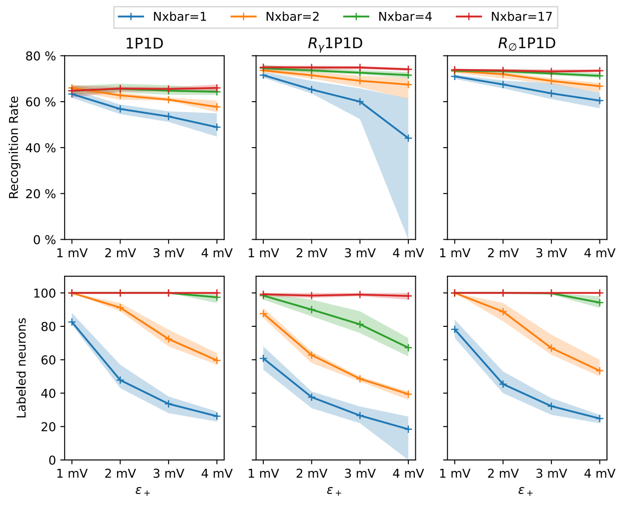


FIGURE 14. Averaged performance and number of labeled neurons after training using one or more crossbars for different values of ϵ chosen randomly in a uniform fashion between 0 mV and ϵ_+. Nxbar indicates how many crossbars were used. For these results, if the simulation ends in a failure (fail-stop), the recognition rate is fixed at 0 % and the number of labeled neurons to 0. Each point (+) represents the average results obtained. The tinted area around each point represents the minimum and maximum values obtained.

According to our results, the higher the upper bound value of the randomly chosen ϵ is, the more detrimental it is when using only one crossbar ($N_{xbar} = 1$), as the number of useful neurons after training drops drastically. Indeed, the higher ϵ is, the more it will prevent the synaptic current from being transmitted to the neuron membrane, because of the parasitic current it causes inside the crossbar. This current is in opposition to the inference current and is not copied by the CCII+. The higher it is, the more synaptic current is needed to cancel it and pass through the CCII+. Because of this, many neurons with a high value of ϵ are not able to charge enough to fire and become useless or "dead". The recognition rate thus drops, but the system can still discriminate thanks to the redundancy of postsynaptic neurons that have lower ϵ values.

Splitting the synaptic array into smaller ones effectively reduces the detrimental impact of the voltage copy offset of the CCII+, as fewer memristors are connected to the same CCII+, thus reducing the adverse current generated. This means that more neurons can effectively learn instead, and the recognition rate drops less.

Interestingly, among the three learning rules, the one that generally handles this voltage copy offset variability best is $R_{\varnothing}$1P1D in terms of performance, despite having a high number of "dead" neurons the higher ϵ can be.

Splitting the crossbar into smaller ones like 68×100 allows us to drastically reduce the detrimental impact of this offset variability. Furthermore, it should also help with the two other problems mentioned before. First, the voltage drop on a postsynaptic line would be greatly reduced as far fewer memristors are in parallel. Secondly, the CCII+ blocks, while much greater in number, would have to handle a smaller current range at the X terminal (from copying a single/a few memristors to handle the inference and write on all the connected memristors). Nevertheless, this solution, while alleviating multiple challenges, needs more CCII+ electronic circuits, thus more space and a higher power consumption.

V. CONCLUSION

In this work we study in greater depth and by simulation a small smart sensor architecture combining a DVS camera and a mixed-mode single-layer SNN with analog neurons and memristive synapses. Looking at the synaptic properties, we show that a small asymmetry in the learning rates in favor of synaptic potentiation can yield superior results, and that such behavior is better handled by the smart sensor compared to an asymmetry in favor of synaptic depression. This is encouraging, because some memristor technologies are already showing such an asymmetry, and it could theoretically be adjusted by adapting the waveforms used to change the synaptic weights. We also show that the architecture is quite resilient to variability over these learning rates around a reference situation. More specifically, having a certain level of synaptic learning rate variability should not cause a complete failure of the sensor. We also looked at the practical initial starting weight distribution, a question usually eluded but critical for an architecture such as ours, which cannot finely tune the weights when the sensor is deployed or will be used for a new application. To tackle this question, we proposed a simple algorithm that makes the SNN learn starting patterns to set it in a known and controlled starting configuration. These initial patterns can have only a fraction of synaptic weights at their high conductive states as long as they are located where the synaptic activity will most likely occur, in order to ensure that the postsynaptic neurons can fire.

We also investigated the impact of some of the parameters of the analog neurons used in the SNN. First, we confirmed that the architecture is resilient to the membrane capacitance variability, in line with some of our previous work with a slightly different configuration, which is a strong point when designing analog postsynaptic neurons. Second, by testing the architecture with different stimulation strengths, we demonstrated the resilience of the architecture in case of smaller synaptic inference currents if one wants to reduce the power consumption or if one is using a synaptic crossbar with shifted conductance ranges.

However, as identified in previous work, this paper confirms that the voltage copy current offset of the CCII is critical, as it generates currents in the crossbar that are detrimental to the behavior of the postsynaptic neurons, especially when the synaptic activity is sparse in time. To alleviate this challenge, among others (voltage drops along the synaptic lines, driving current needed per postsynaptic neuron), we show that splitting the synaptic crossbar into smaller ones is a viable solution, while coming with the drawback of the need for more electronic circuits.

Overall, among the three learning rules, the weakly supervised $R\varnothing$1P1D is the one that shows the best resilience while potentially consuming less current in a learning phase.

This work provides indications for the design and manufacture of an intelligent visual sensor capable of learning on the fly. Current performances could be improved, based on the conclusions of this paper, by studying an SNN with hidden layers. This future network will have to keep in mind technologically resilient solutions that can be easily implemented in hardware, to reduce silicon surface area and power consumption.

APPENDIX A
METHODS

All the simulations presented in this work are obtained using a custom simulator written in Python to estimate the behavior of the system. The computation method of the simulator is briefly described here.

1) COMPUTING THE POSTSYNAPTIC CURRENTS

To compute the synaptic currents two different methods are used depending on the value of ϵ. If the input offsets ϵ of all postsynaptic neurons are equal to 0, the postsynaptic currents injected into the membrane (i_{stim}) are acquired by a vector-matrix product between the synaptic weights matrix of the layer and the vector of presynaptic voltages. Otherwise, if each postsynaptic has different value of ϵ, the computation of synaptic current is done by a Hadamard product between the matrix of each synaptic voltage ($V_{pre\ i,j} - V_{post\ i,j}$) and the matrix of the synaptic weights to get all the synaptic currents. They are then summed along the postsynaptic lines to get the current collected by the postsynaptic neurons.

2) COMPUTING THE MEMBRANE VOLTAGES

As we only have square waveforms or continuous voltages used to generate currents that will be injected into the postsynaptic membranes, the evolution of the membrane voltage can be described by a slope. Thus, to compute the increase or decrease of the membrane voltage (slope) until it reaches the threshold, we generate 2 computation points: one at the start of the presynaptic waveform at time t; one at the end at time $t+T_{LTP}$. As the stimulation current i_{stim} is constant between those two computation points per presynaptic events (as is the discharge current for positive membrane voltages), we can compute the membrane voltage as shown in Fig. 15. The equation used to compute the membrane voltage increase $\Delta V_{membrane}$ between two computation points ($t_j - 1$ and t_j) is given by Equation (2), where $\Delta T = t_j - t_{j-1}$ and $i_{stim} = K \cdot i_X$ if $i_X > 0$ else $i_{stim} = 0\,\text{A}$. Furthermore, the membrane voltage is clipped between 0 V and the operating voltage (5 V in our simulations).

$$\Delta V_{\text{membrane}}|_{\Delta T} = \underbrace{\frac{i_{stim}}{C_{\text{mem}}}\Delta T}_{\text{synaptic charge}} - \underbrace{\frac{i_{\text{discharge}}}{C_{\text{mem}}}\Delta T}_{\text{membrane leak}} \quad (2)$$

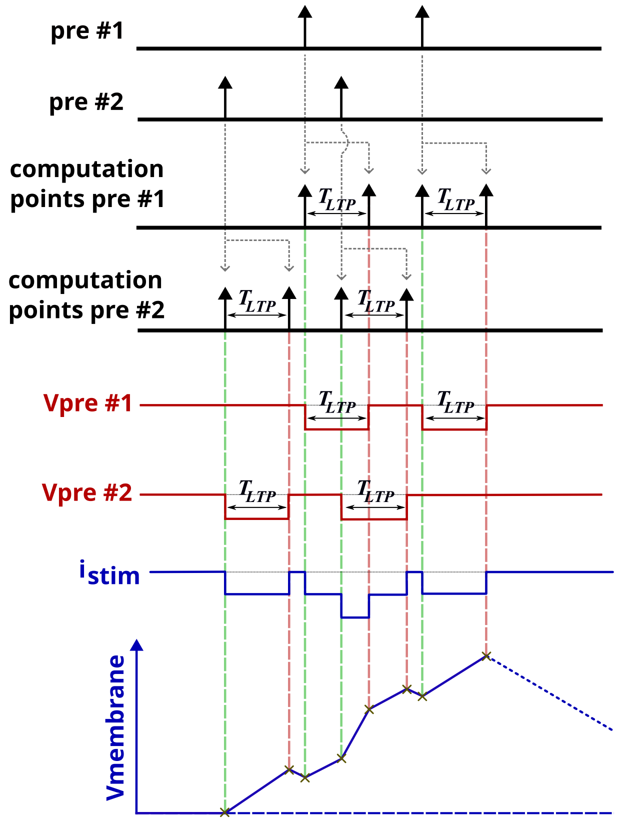


FIGURE 15. Computation method of the membrane voltage when events occur at 2 different presynaptic inputs. Two computation points are generated per synaptic event and, because the current is constant between computation points, the membrane capacitance can be computed by finding the corresponding membrane voltage slope.

3) DETERMINING WHICH POSTSYNAPTIC NEURON FIRED

If at a computation point, at least one postsynaptic membrane surpassed the membrane threshold ($V_{threshold}$), we compute the instant where it was reached per postsynaptic neuron that crossed the threshold by using the computed slope of the membrane voltage evolution between the two computation points. The system will choose the postsynaptic neuron according to a slightly modified Winner Take All (WTA) rule that keeps the postsynaptic neuron that fired first. In our system, the DCB (see Fig. 2) controlling all the postsynaptic neurons works with a fixed clock frequency to save which neuron fired before resetting the others. Thus, if more than one postsynaptic neuron fired between two computational clock edges of the DCB, we used an arbiter that should be located in the Digital Control Block to keep only one output neuron as illustrated in Fig. 16. This arbiter, working at a clock T_{clk} chose the output neuron with the lowest index if more than one are detected between two computational clock edges.

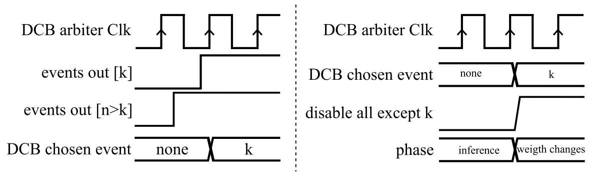


FIGURE 16. Output event arbiter inside the DCB to select only one postsynaptic neuron if more than one fired between two clock computational edges. 'events out' represents the output events received by the DCB. 'DCB chosen' event shows which postsynaptic is chosen.

4) RECOGNITION RATE

The recognition rate (RR) used for the results reported in this paper is computed after the test phase using the following equation:

$$RR = \frac{\text{Number of correct answers}}{N_{\text{test}}}$$

APPENDIX B
REFERENCE SITUATION PARAMETERS

Table 5 gives the reference parameters of the simulated SNN.

TABLE 5. Reference parameters of the spiking neural network.

Parameter	Value	Description
N_{inputs}	1156	Amount of presynaptic neurons (34×34)
N_{outputs}	100	Amount of postsynaptic neurons
$\epsilon+$	0 mV	CCII+ voltage copy offset upper bound
$V_{\text{threshold}}$	1 V	Membrane threshold voltage
K	0.01	CCII+ scaling factor
N_{refrac}	10	Refractory counter period value
C_{mem}	1 pF	Membrane capacitance
Δv_{stim}	1 V	Absolute voltage for inference
$i_{\text{discharge}}$	100 pA	Membrane leakage current
V_{thpot}	1.2 V	Voltage threshold for potentiation *
V_{thdep}	-1.2 V	Voltage threshold for depression *
T_{LTP}	10 µs	Inference pulse width
A_{pot}	10 %	Potentiation learning rate
A_{dep}	10 %	Depression learning rate
G_{max}	1 µS	Memristor conductance upper bound
G_{min}	10 nS	Memristor conductance lower bound
T_{clk}	1 µs	Output arbiter clock period

* Values indicated as an example.

APPENDIX C
PRESYNAPTIC NEURONS MODEL

To infer data through the SNN, the presynaptic neurons only apply a negative square waveform with a width T_{LTP} and absolute amplitude Δv_{stim} that corresponds to an input event. The inference waveform in this architecture is negative for two reasons:

- The first reason is hereditary and linked to the original analog implementation of the STDP using a CCII+ [41].

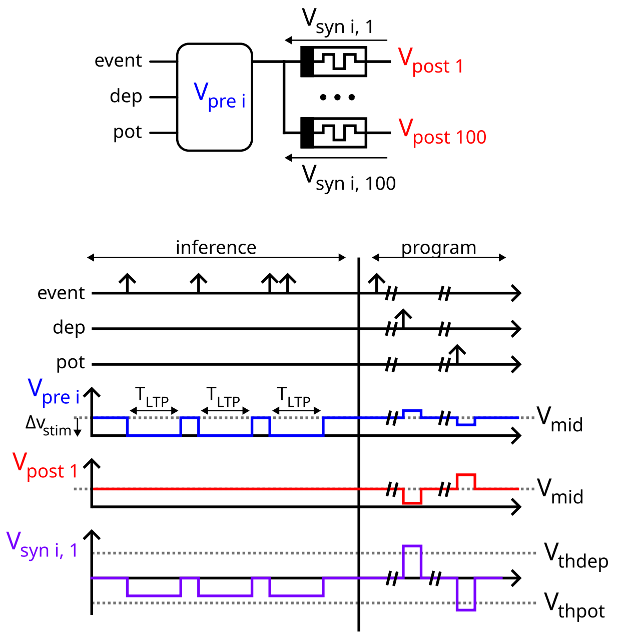


FIGURE 17. Presynaptic neuron model and example of the waveforms that it can apply on the connected memristors. A postsynaptic waveform is also shown as an example.

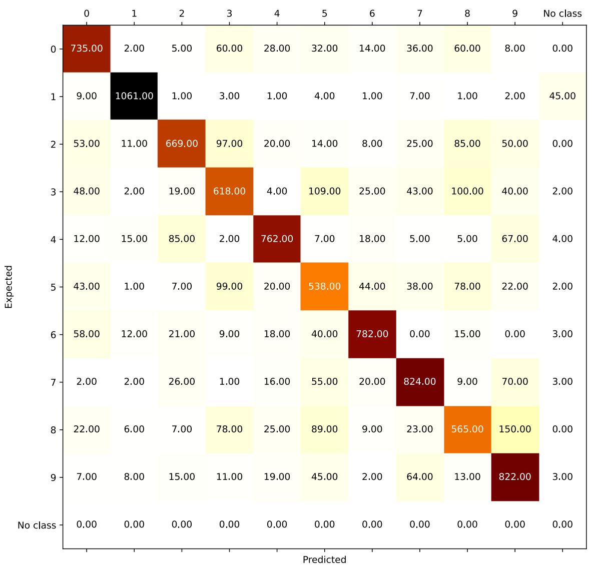
- The second reason is due to the CCII+ normalization. To have a positive current injected in the neuron membrane, the current i_X must be pumped from the CCII+ X terminal, thus leading to the need for a negative current flowing through the memristor due to normalization.

This negative piecewise waveform needs to be long enough to generate current to stimulate the postsynaptic membrane while being short enough not to consume too much power. It also needs to be short enough to allow the application of successive events of the same input as in our architecture; while a presynaptic input is active (applying the inference waveform) it cannot apply a new inference waveform on the same input. Furthermore, a T_{LTP} that is too high would also mean the need for higher time constants to generate them in the analog neurons, most likely leading to the use of a greater surface if capacitors are used to generate them. Considering all of this, we choose to run our simulation with a T_{LTP} of $10\,\mu$s, which is of the order of magnitude of time constants in microelectronics. As this value of T_{LTP} is used to define some of the following parameters, if a smaller T_{LTP} is used, the other neuron parameters should be adjusted accordingly. The value of the absolute amplitude Δv_{stim} is fixed to 1 V assuming that it is below the voltage needed to change the memristives weights. In our case, as it is negative, it needs to be lower than V_{thpot}, the negative threshold voltage (to increase a synaptic conductance). If it is not, this value can be adjusted by dividing K by this factor to keep the current stimulation of the membrane identical.

As the presynaptic neurons have the charge of applying the inference waveform and part of the programming (weight changing) waveforms used to modify a weight, they need to be able to generate 3 different waveforms, as illustrated in Fig. 17: one when an event is received; one to potentiate the synapse and one to depreciate the synapse.

APPENDIX D
CONFUSION MATRIX AND CONDUCTANCE MAPS IN THE REFERENCE SITUATION

Fig. 18 shows the raw confusion matrix obtained for one simulation in the reference situation obtained with the rule $R_{\varnothing}$1P1D. We can see that some test samples did not permit the SNN to fire (represented by No class column).



Expected \ Predicted	0	1	2	3	4	5	6	7	8	9	No class
0	735.00	2.00	5.00	60.00	28.00	32.00	14.00	36.00	60.00	8.00	0.00
1	9.00	1061.00	1.00	3.00	1.00	4.00	1.00	7.00	1.00	2.00	45.00
2	53.00	11.00	669.00	97.00	20.00	14.00	8.00	25.00	85.00	50.00	0.00
3	48.00	2.00	19.00	618.00	4.00	109.00	25.00	43.00	100.00	40.00	2.00
4	12.00	15.00	85.00	2.00	762.00	7.00	18.00	5.00	5.00	67.00	4.00
5	43.00	1.00	7.00	99.00	20.00	538.00	44.00	38.00	78.00	22.00	2.00
6	58.00	12.00	21.00	9.00	18.00	40.00	782.00	0.00	15.00	0.00	3.00
7	2.00	2.00	26.00	1.00	16.00	55.00	20.00	824.00	9.00	70.00	3.00
8	22.00	6.00	7.00	78.00	25.00	89.00	9.00	23.00	565.00	150.00	0.00
9	7.00	8.00	15.00	11.00	19.00	45.00	2.00	64.00	13.00	822.00	3.00
No class	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

FIGURE 18. Raw confusion matrix obtained after the simulation of the reference situation for the same initial synaptic weights. The lines correspond to the sample class (expected). The columns correspond to the SNN output (Predicted). The No class column corresponds to the case were the SNN did not fire. The No class line corresponds to test samples without a label (none in the used dataset).

REFERENCES

[1] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, "The history began from AlexNet: A comprehensive survey on deep learning approaches," 2018, *arXiv:1803.01164*.

[2] S. Dodge and L. Karam, "Human and DNN classification performance on images with quality distortions: A comparative study," *ACM Trans. Appl. Perception*, vol. 16, no. 2, pp. 1–17, Apr. 2019, doi: 10.1145/3306241.

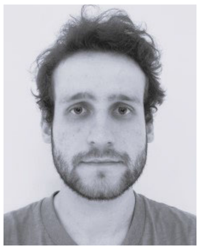
[3] S. Dodge and L. Karam, "A study and comparison of human and deep learning recognition performance under visual distortions," in *Proc. 26th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Vancouver, BC, Canada, Jul. 2017, pp. 1–7, doi: 10.1109/ICCCN.2017.8038465.

[4] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958, doi: 10.1037/h0042519.

[5] K.-S. Oh and K. Jung, "GPU implementation of neural networks," *Pattern Recognit.*, vol. 37, no. 6, pp. 1311–1314, Jun. 2004, doi: 10.1016/j.patcog.2004.01.013.

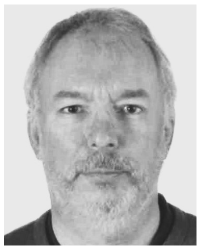
[6] N. Jouppi, C. Young, N. Patil, and D. Patterson, "Motivation for and evaluation of the first tensor processing unit," *IEEE Micro*, vol. 38, no. 3, pp. 10–19, May 2018, doi: 10.1109/MM.2018.032271057.

[7] S. Rivas-Gomez, A. J. Pena, D. Moloney, E. Laure, and S. Markidis, "Exploring the vision processing unit as co-processor for inference," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, Vancouver, BC, Canada, May 2018, pp. 589–598, doi: 10.1109/IPDPSW.2018.00098.
[8] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, 1997, doi: 10.1016/S0893-6080(97)00011-7.
[9] 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 Neurosci.*, vol. 13, p. 625, Jul. 2019, doi: 10.3389/fnins.2019.00625.
[10] M. Stimberg, R. Brette, and D. F. Goodman, "Brian 2, an intuitive and efficient neural simulator," *eLife*, vol. 8, Aug. 2019, Art. no. e47314, doi: 10.7554/eLife.47314.
[11] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers Neurosci.*, vol. 11, p. 682, Dec. 2017, doi: 10.3389/fnins.2017.00682.
[12] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 $\times$ 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008, doi: 10.1109/JSSC.2007.914337.
[13] H.-P. Cheng, W. Wen, C. Wu, S. Li, H. H. Li, and Y. Chen, "Understanding the design of IBM neurosynaptic system and its tradeoffs: A user perspective," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Lausanne, Switzerland, Mar. 2017, pp. 139–144, doi: 10.23919/DATE.2017.7926972.
[14] J. Hsu, "IBM's new brain [news]," *IEEE Spectr.*, vol. 51, no. 10, pp. 17–19, Oct. 2014, doi: 10.1109/MSPEC.2014.6905473.
[15] M. Davies et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018, doi: 10.1109/MM.2018.112130359.
[16] C. S. Thakur, J. L. Molin, G. Cauwenberghs, G. Indiveri, K. Kumar, N. Qiao, J. Schemmel, R. Wang, E. Chicca, J. O. Hasler, J.-S. Seo, S. Yu, Y. Cao, A. van Schaik, and R. Etienne-Cummings, "Large-scale neuromorphic spiking array processors: A quest to mimic the brain," *Frontiers Neurosci.*, vol. 12, p. 891, Dec. 2018, doi: 10.3389/fnins.2018.00891.
[17] S. Schmitt et al., "Neuromorphic hardware in the loop: Training a deep spiking network on the BrainScaleS wafer-scale system," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Anchorage, AK, USA, May 2017, pp. 2227–2234, doi: 10.1109/IJCNN.2017.7966125.
[18] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proc. IEEE*, vol. 102, no. 5, pp. 699–716, May 2014, doi: 10.1109/JPROC.2014.2313565.
[19] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the SpiNNaker system architecture," *IEEE Trans. Comput.*, vol. 62, no. 12, pp. 2454–2467, Dec. 2013, doi: 10.1109/TC.2012.142.
[20] Y. Tang, J. R. Nyengaard, D. M. G. De Groot, and H. J. G. Gundersen, "Total regional and global number of synapses in the human brain neocortex," *Synapse*, vol. 41, no. 3, pp. 258–273, Sep. 2001, doi: 10.1002/syn.1083.
[21] P. Lennie, "The cost of cortical computation," *Current Biol.*, vol. 13, no. 6, pp. 493–497, Mar. 2003, doi: 10.1016/S0960-9822(03)00135-0.
[22] W. B. Levy and V. G. Calvert, "Communication consumes 35 times more energy than computation in the human cortex, but both costs are needed to predict synapse number," *Proc. Nat. Acad. Sci. USA*, vol. 118, no. 18, May 2021, Art. no. e2008173118, doi: 10.1073/pnas.2008173118.
[23] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses," *Frontiers Neurosci.*, vol. 9, p. 141, Apr. 2015, doi: 10.3389/fnins.2015.00141.
[24] S. J. Kim, S. Kim, and H. W. Jang, "Competing memristors for brain-inspired computing," *iScience*, vol. 24, no. 1, Jan. 2021, Art. no. 101889, doi: 10.1016/j.isci.2020.101889.
[25] A. Chanthbouala, V. Garcia, R. O. Cherifi, K. Bouzehouane, S. Fusil, X. Moya, S. Xavier, H. Yamada, C. Deranlot, N. D. Mathur, M. Bibes, A. Barthélémy, and J. Grollier, "A ferroelectric memristor," *Nature Mater.*, vol. 11, no. 10, pp. 860–864, Oct. 2012, doi: 10.1038/nmat3415.
[26] S. Boyn, S. Girod, V. Garcia, S. Fusil, S. Xavier, C. Deranlot, H. Yamada, C. Carrétéro, E. Jacquet, M. Bibes, A. Barthélémy, and J. Grollier, "High-performance ferroelectric memory based on fully patterned tunnel junctions," *Appl. Phys. Lett.*, vol. 104, no. 5, Feb. 2014, Art. no. 052909, doi: 10.1063/1.4864100.
[27] X. Ji, Z. Dong, Y. Han, C. S. Lai, G. Zhou, and D. Qi, "EMSN: An energy-efficient memristive sequencer network for human emotion classification in mental health monitoring," *IEEE Trans. Consum. Electron.*, p. 1, 2023, doi: 10.1109/TCE.2023.3263672.
[28] Z. Dong, X. Ji, G. Zhou, M. Gao, and D. Qi, "Multimodal neuromorphic sensory-processing system with memristor circuits for smart home applications," *IEEE Trans. Ind. Appl.*, vol. 59, no. 1, pp. 47–58, Jan. 2023, doi: 10.1109/TIA.2022.3188749.
[29] P. Lewden, A. F. Vincent, C. Meyer, J. Tomas, S. Siami, and S. Saïghi, "Hardware spiking neural networks: Slow tasks resilient learning with longer term-memory bits," in *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, Nara, Japan, Oct. 2019, pp. 1–4, doi: 10.1109/BIOCAS.2019.8918992.
[30] P. Lewden, A. F. Vincent, C. Meyer, J. Tomas, and S. Sïghi, "Toward hardware spiking neural networks with mixed-signal event-based learning rules," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Glasgow, U.K., Jul. 2020, pp. 1–8, doi: 10.1109/IJCNN48605.2020.9206736.
[31] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers Neurosci.*, vol. 9, p. 437, Nov. 2015, doi: 10.3389/fnins.2015.00437.
[32] L. R. Iyer and A. Basu, "Unsupervised learning of event-based image recordings using spike-timing-dependent plasticity," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Anchorage, AK, USA, May 2017, pp. 1840–1846, doi: 10.1109/IJCNN.2017.7966074.
[33] T. Masquelier and S. J. Thorpe, "Unsupervised learning of visual features through spike timing dependent plasticity," *PLoS Comput. Biol.*, vol. 3, no. 2, p. e31, Feb. 2007, doi: 10.1371/journal.pcbi.0030031.
[34] 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 Recognit.*, vol. 94, pp. 87–95, Oct. 2019, doi: 10.1016/j.patcog.2019.05.015.
[35] M. Mozafari, S. R. Kheradpisheh, T. Masquelier, A. Nowzari-Dalini, and M. Ganjtabesh, "First-spike-based visual categorization using reward-modulated STDP," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 6178–6190, Dec. 2018, doi: 10.1109/TNNLS.2018.2826721.
[36] S. Majumdar, H. Tan, Q. H. Qin, and S. van Dijken, "Energy-Efficient organic ferroelectric tunnel junction memristors for neuromorphic computing," *Adv. Electron. Mater.*, vol. 5, no. 3, Mar. 2019, Art. no. 1800795, doi: 10.1002/aelm.201800795.
[37] S. Boyn, J. Grollier, G. Lecerf, B. Xu, N. Locatelli, S. Fusil, S. Girod, C. Carrétéro, K. Garcia, S. Xavier, J. Tomas, L. Bellaiche, M. Bibes, A. Barthélémy, S. Saïghi, and V. Garcia, "Learning through ferroelectric domain dynamics in solid-state synapses," *Nature Commun.*, vol. 8, no. 1, Apr. 2017, Art. no. 14736, doi: 10.1038/ncomms14736.
[38] C. Ma, Z. Luo, W. Huang, L. Zhao, Q. Chen, Y. Lin, X. Liu, Z. Chen, C. Liu, H. Sun, X. Jin, Y. Yin, and X. Li, "Sub-nanosecond memristor based on ferroelectric tunnel junction," *Nature Commun.*, vol. 11, no. 1, p. 1439, Mar. 2020, doi: 10.1038/s41467-020-15249-1.
[39] D. Querlioz, O. Bichler, and C. Gamrat, "Simulation of a memristor-based spiking neural network immune to device variations," in *Proc. Int. Joint Conf. Neural Netw.*, San Jose, CA, USA, Jul. 2011, pp. 1775–1781, doi: 10.1109/IJCNN.2011.6033439.
[40] E. M. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1063–1070, Sep. 2004, doi: 10.1109/TNN.2004.832719.
[41] G. Lecerf, J. Tomas, and S. Saïghi, "Excitatory and inhibitory memristive synapses for spiking neural networks," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Beijing, China, May 2013, pp. 1616–1619, doi: 10.1109/ISCAS.2013.6572171.
[42] C. Meyer, "Conception de réseaux de neurones sur silicium à l'aide de synapses memristives: Application Au traitement d'image," M.S. thesis, Dept. Electronique, Université Bordeaux, Bordeaux, France, 2021.
[43] M. R. Mahmoodi, A. F. Vincent, H. Nili, and D. B. Strukov, "Intrinsic bounds for computing precision in memristor-based vector-by-matrix multipliers," *IEEE Trans. Nanotechnol.*, vol. 19, pp. 429–435, 2020, doi: 10.1109/TNANO.2020.2992493.



PIERRE LEWDEN received the Ph.D. degree in electronics on hardware implementation of event-based neural networks based on nanotechnologies from the University of Bordeaux, France, in 2023. He is currently a Research Fellow with the DesCartes Research Program, CNRS@CREATE, Singapore, where he is also working on the implementation of spiking neural networks on FPGAs.

ADRIEN F. VINCENT received the Ph.D. degree in the use of innovative memory devices as artificial synapses in neuro-inspired electronics from the University of Paris-Saclay, France, in 2017. After graduating, he was a Postdoctoral Researcher with the University of California at Santa Barbara, Santa Barbara, USA, in the group of Prof. Dmitri Strukov, working on the statistical modeling of TiO2 memristors and their potential for analog dot-product engines. Since 2018, he has been an Associate Professor with Bordeaux INP. He has joined the Bioelectronics Group, IMS Laboratory, Talence, France, where he works on hardware implementation of artificial intelligence systems and neuromorphic architectures.



JEAN TOMAS received the Diploma degree in electrical engineering from Ecole Nationale Supérieure d'Electronique et de Radioélectricité de Bordeaux (ENSERB), in 1985, and the Ph.D. degree in electrical engineering from Université Bordeaux 1, in 1988. Currently, he is an Associate Professor with the IMS Laboratory (UMR 5218 CNRS), Université de Bordeaux. His research interests include design of analog and mixed signal circuits and systems dedicated to neuromorphic applications, such as memristive spiking neural network for edge computing.

SYLVAIN SAÏGHI received the Ph.D. degree in the design of analog operators dedicated to silicon neurons, in 2004. He is currently a Full Professor with the University of Bordeaux. He has performed pioneering work in developing biologically realistic and tunable silicon neurons. He has also authored or coauthored more than 60 peer-review publications. Thanks to a Fulbright Scholar Grant, he was a Visiting Associate Professor with Johns Hopkins University, Baltimore, MD, USA, for six months, in 2011. His current research interest includes the hardware implementation of neuromorphic systems for edge computing, with projects ranging from co-integrating emerging memristive nanodevices with CMOS circuits to strategies relying on more conventional digital systems.