Parameter Exploration to Improve Performance of Memristor-Based Neuromorphic Architectures

Mahyar Shahsavari, Pierre Boulet

Parameter Exploration to Improve Performance of Memristor-Based Neuromorphic Architectures

Mahyar Shahsavari, Pierre Boulet, *Member, IEEE*

Abstract—The brain-inspired spiking neural network neuromorphic architecture offers a promising solution for a wide set of cognitive computation tasks at a very low power consumption. Due to the practical feasibility of hardware implementation, we present a memristor-based model of hardware spiking neural networks which we simulate with N2S3 (Neural Network Scalable Spiking Simulator), our open source neuromorphic architecture simulator. Although Spiking neural networks are widely used in the community of computational neuroscience and neuromorphic computation, there is still a need for research on the methods to choose the optimum parameters for better recognition efficiency. With the help of our simulator, we analyze and evaluate the impact of different parameters such as number of neurons, STDP window, neuron threshold, distribution of input spikes and memristor model parameters on the MNIST hand-written digit recognition problem. We show that a careful choice of a few parameters (number of neurons, kind of synapse, STDP window and neuron threshold) can significantly improve the recognition rate on this benchmark (around 15 points of improvement for the number of neurons, a few points for the others) with a variability of 4 to 5 points of recognition rate due to the random initialization of the synaptic weights.

Index Terms—Neuromorphic Computing, Parameter Evaluations, Spiking Neural Networks, Memristor, Unsupervised Learning

1 Introduction

Neuromorphic computing has the potential to bring very low power computation to future computer architectures and embedded systems [1]. Indeed parallel neuromorphic computing, by doing computation and storage in the same devices can overcome the Von-Neumann bottleneck which is coming from speed difference between the processor and the memory. There are two broad types of brain-inspired cognitive computations: abstract artificial neural networks (ANNs) and closer to biology spiking neural networks (SNNs) [2]. Machine learning algorithms such as classical ANNs and more recently deep belief networks [3], [4] are widely used for data classification and clustering. However, ANNs are a highly abstract mathematical model of neurons that are designed to be executed on digital processing platforms (more and more using accelerating units such as GPUs). There are two different types of data coding: *rate* and *spike*. In the rate coding, the rate of each data is important, while in spike coding the spikes are the same and the *timing* of spikes is important. ANNs only use rate coding to represent neuronal activity and are not capable of taking into account the precise relative timing of spikes. In contrast, timing in SNNs is significant which make them suitable for dealing with natural signals in real-time that are getting more and more significant in the internet of things using online learning. The data should be coded to spikes to be processed in SNN, which is known as spike coding (versus rate coding in ANN). SNNs offer online real time unsupervised learning through continuous weight updating which is performed on local synaptic weights. This temporal and spatial locality is important in hardware implementations of neural networks because it frees this architecture of the memory bottleneck of Von Neumann architectures.

• *M. Shahsavari and P. Boulet are with Univ. Lille, CNRS, Centrale Lille, UMR 9189 – CRIStAL – Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France.*

Recent advances in nanotechnology have provided neuromorphic computing architecture with novel memristive devices which have the capability of mimicking synaptic plasticity, such as resistive switching memory (RRAM) [5], [6], [7], phase change memory (PCM) [8], [9], [10], conductive bridge memory (CBRAM) [11], [12], [13], and ferroelectric memory (FeRAM) [14], [15]. The advantages of using these memristive nanodevices to model the behavior of synapses are their unique properties, such as scalability, flexibility because of their analog behavior, manufacturability on top of CMOS technology to make a crossbar array (shown in Figure 1) and ability to remember the last state in a SNN [16]. Fortunately due to close collaboration with the nano-electronics research center in the University of Lille (IEMN), we have the opportunity to study the appropriateness of various classes of memristors (e.g.,TiO_2, NOMFET, Magnetoelectric) to build a SNN hardware platform by using real parameters. In order not to restrict our study of parameter exploration to only one device, we have chosen here the resistive memory model presented in [17] which is a generic memristive device model.

It is widely believed that plasticity is the key of learning in the biological brain [18]. Consequently, with the latest proposals to use the memristive nano-devices as synapses, we implement an efficient and well-studied unsupervised learning rule known as spike timing dependent plasticity (STDP) [19], [20]. In this study, building upon our previous work [21], we show that the memristive synapses are adapted to unsupervised STDP learning in SNNs. We explain the required technology and architecture with system-level simulation. For implementation and results, we use the MNIST dataset of handwritten digits [22] to test the performance of neural networks. Although, there are various research works using STDP learning in SNN architecture and neuromorphic VLSI implementations, none of them evaluates and analyzes the key parameters that improve learning and SNN recognition performance in those architectures. Our main contribution is to evaluate and explore the impact of several parameters on the learning performance of SNNs for the

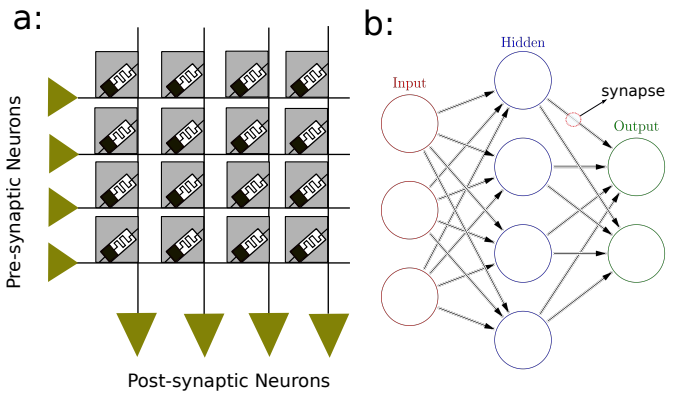

Fig. 1. Crossbar arrays allow to build SNNs. **a)** The memristive synapse connects the spiking neurons in configurable crossbar array suitable for STDP unsupervised learning, the presynaptic neurons are considered as inputs and postsynaptic neurons as outputs. **b)** Two crossbar arrays can interconnect the neurons of a three-layer feed-forward SNN.

MNIST benchmark:

- the number of neurons in the output layer,
- different distributions of spikes to code the input images,
- the model of the synapses (this parameter alone has been studied in our previous work [21]),
- the duration of the STDP window,
- various thresholds for adaptive threshold LIF neurons,
- and the memristive synapse fitting parameter.

We find that the most significant parameters are the number of neurons (from less than 70 % of recognition rate with 20 neurons to around 85 % with 100 neurons), the model of synapse (difference of a few points of recognition rate in average between the two tested models) and its properties and the neuron threshold (a few points of improvement).

A lot of efforts have been put into developing appropriate simulation tools and techniques [12], [23]. In this paper, for implementation we use our simulator, N2S3 (Neural Network Scalable Spiking Simulator), an open source event-driven simulator, that is dedicated to the architecture exploration of hardware SNNs.

In the following, we first survey the hardware spiking neural networks systems, then present our simulation tool, N2S3, that we use in the following experimental evaluation of the various parameters of SNNs with the MNIST benchmark.

2 Hardware Spiking Neural Network System Survey

Specific application domains such as Big Data classification, visual processing, pattern recognition and in general sensory input data, require information processing systems which are able to classify the data and to learn from the patterns in the data. Such systems should be power-efficient. Thus researchers have developed brain-inspired architectures such as spiking neural networks. For large scale brain-like computing on neuromorphic hardware, there are four approaches:

1) Microprocessor based approaches where the system can read the codes to execute and model the behavior of neural systems and cognitive computation such as the SpiNNaker machine [24].
2) Fully digital custom circuits where the neural system components are modeled in circuit using sate-of-the-art CMOS technology e.g., IBM TrueNorth machine [1].
3) Analog/digital mixed-signal systems that model the behavior of biological neural systems, e.g. the Neurogrid [25] and BrainScales [26] projects.
4) Memristor crossbar array based systems where the analog behavior of the memristors emulate the synapses of a spiking neural network.

In the following, we give some details about these approaches and compare their performance. Because we used MNIST for experimental evaluation, we present the results of classification for SpiNNaker, TrueNorth and BrainScales. We did not find the same evaluation on Neurogrid.

SpiNNaker is a massively parallel and processor-based (ARM processor) system with the purpose of building large scale spiking neural networks simulations. It is highly scalable and capable to simulate a network from thousands to millions of neurons with varying degree of connectivity. It proposes to integrate 57,600 custom VLSI chips based on the AER (Address Event Representation) communication protocol [27]. Each chip Contains 18 fixed-point advanced RISC ARM968 processing cores next to the custom routing infrastructure circuits which is dedicated 96 KB of local memory besides 128 MB of shared Dynamic Random Access Memory (DRAM) as it is depicted in Figure 2.a. The router memory consists of a three-state 1024×32 bit Content Addressable Memory (CAM) and a 1024×24 bit Random Access Memory (RAM). Going more to the details, each ARM core has a local 32 KB instruction memory and 64 KB data memory. Regarding to the architecture and design properties, SpiNNaker offers very fast simulation of large scale neural networks. It has a remarkable flexibility for arbitrary connectivity for network architecture and various neurons, synapses and learning algorithms. However, the system still uses Von Neumann architecture with a large extent of memory hierarchies found in conventional computers with memory wall bottleneck issues. Because of using low-power ARM processors dedicated to power-efficient platforms used in training and robotic applications with four to 48 nodes, SpiNNaker consumes a relatively small amount of power. Testing MNIST on SpiNNaker has been reported in [28] using spike-based variation of previously trained Deep Belief Networks (DBNs). To implement spiking DBNs on SpiNNaker, a collection of functions were developed in Python that read a MATLAB file of an off-line trained DBN and generate a description of network ready to run on SpiNNaker. The classification accuracy for this work is 95 %, which consumes only 0.3 W with classification latencies in the order of tens of milliseconds. However, there is yet the memory wall issues of conventional computing platforms as SpiNNaker has Von Neumann architecture. For better weight resolution 22 bits are applied to store network weights that can be replaced only by one memristor in non-Von Neumann neuromorphic platforms. Among the parameters we explore in our work, neuron firing threshold, spike distribution and number of neurons could be applicable for future similar research using SpiNNaker platform.

IBM designed a scalable, flexible and non-Von Neumann full custom spiking neural network named “TrueNorth”. Although TrueNorth uses transistors and digital gates, but they use event-driven method to communicate in fully asynchronous manner. The structure of TrueNorth consists of 5.4 billion transistors to build 4096 neurosynaptic cores. Each core includes 256 digital

LIF neuron, 256×256 binary programmable synapses, and asynchronous encoding/decoding and routing circuits. Each synapse has binary behavior that can be individually turned on or off and can be assigned to model one type of inhibitory and two types of excitatory synapse with different weights. Neuron dynamics has a global 1 kHz clock and so is discretized by into 1 ms time steps. Regarding to the synaptic matrix, each neuron can be connected to one up to 256 neurons of a destination core. The routing in TrueNorth is less flexible than in SpiNNaker, however TrueNorth can distribute the system memory includes core synaptic matrix and routing table entries (Figure 2.b). The architecture thus supports dynamics of connectivity that includes feed-forward, recurrent, and lateral connections. In this platform the synapses do not implement any plasticity mechanism, therefore they are not able to perform on-line learning. TrueNorth is used to recognize and classify MNIST dataset images using unsupervised learning in RBM architecture [29]. The system scored 91.94 % using 60,000 samples for training set and 10,000 samples of test set. As the TrueNorth is a non-Von Neumann architecture using spikes, parameters we are evaluating in this study are applicable here more than to the SpiNNaker platform. Memristor could be an alternative for synapse model in future development of TrueNorth as the architecture has neurons next to synapses.

The BrainScales project (Brain-inspired multiscale computation in neuromorphic hybrid systems) is the successor of FACETS [30] project. This project proposes the design and implementation of a custom analog/digital mixed-signal simulation engine that is able to implement the differential equations with an acceptable accuracy. This computational neuroscience model is provided by neuroscientists, and reproduces the results obtained from numerical simulations executed on conventional computers. The Heidelberg University BrainScales project (HICANN chip) aims to produce a wafer-scale neural simulation platform, in which each 8 inch silicon wafer integrates 50×106 plastic synapses and 200,000 biologically realistic neuron circuits (see Figure 2.c). In order to have a scalable size with maximum number of processors on the wafer, relatively small capacitors have been applied for modeling the synapses and neurons. Accordingly, using the large currents generated by the above-threshold circuit and the small capacitors, the BrainScales circuits is not able to achieve the long time-constants required for interacting with the real-time environments. However, the speed of network components operations compared to biological elements reactions is accelerated by a factor of 10^3 or 10^4 which can reduce the simulation time dramatically. Furthermore, it needs large bandwidth and fast switching and still high-power circuit for propagating spikes across the network [31]. PyNN is a Python-based language designed for describing spiking neural network models [32]. The integration of the operating software framework for the BrainScales (FACETS) is now available in PyNN. It represents an optimal way to provide a convenient interface to work with BrainScales neuromorphic devices that allows to benchmark and verify the hardware model. MNIST could be a good experimental use-case to compare the performance of BrainScales with other neuromorphic platforms. Additionally, regarding the potential of memristor nanodevice, we recommend it to play the role of synapse in BrainScales architecture. The model of memristor could be a similar model that we are using in this work.

Neurogrid is another big project is developed at Stanford University that emulates neuromorphic engineering vision, sub-threshold network components circuits and uses analog/digital mixed-signal to model continuous time for network elements. This neuromorphic platform simulates a million neurons with billions of synaptic connections in real-time. Similar to TrueNorth and BrainScales the architecture of Neurogrid is non-Von Neumann. Neurogrid emulates four network elements, axon, dendrite, soma and synapse. Only the axon circuit is digital and the other elements are modeled in the analog circuits due to the better energy efficiency. Neurogrid consists of 16 standard CMOS "NeuroCores" (see Figure 2.d) integrated on a board that works using 3 W of power energy connected in a tree network, with each NeuroCore consisting of a 256×256 array of two-compartmental neurons. The synaptic circuits are shared among the neurons while different spikes can be assigned to the same synapse. The main goal of neuromorphic systems is to interact with real physical environments and process the natural signals with physiological time-scales, Neurogrid has long time constants in the range of tens of milliseconds. Consequently, this long time constants limitation causes difficulty in using typical VLSI for design and implementation. Neurogrid and BrainScales similarly use the temporal dynamic of memory elements to store the state of the network. Accordingly, these two projects have the capability of local learning using the STDP learning rule. NGPython is a user interface software allows a user to specify neuronal models in the Python programming environment. As computational elements in Neurogrid and BrainScales such as using spiking data, model of neurons, Non-Von Neumann architecture, synaptic connections and STDP local learning are similar to our platform, N2S3 [33], [34] could be a potential software tools for simulation the network elements and even an interface between hardware and users. The scalability to be used in distributed systems, flexibility and suing memristor nanodevice as synaptic connections in N2S3, provide the possibility of simulating hardware models using memristive synapses beside other already existed elements in Neurogrid or BrainScales.

Yet there is another alternative in addition to mentioned architecture that has been proposed by several authors [5], [16], [35], [36], [37], using memristive devices as synapses in neuromorphic circuits. This has the potential to lower the energy consumption by a large proportion with the ability of storing the network synaptic weights for long term in a low cost way. It has also been showed that the memristors can emulate the STDP learning rule, and thus lead to unsupervised learning circuits. Therefore, this neuromorphic architecture can learn on-line. Additionally, using memristive synapse next to CMOS neuron cause a non-Von Neumann architecture and more efficient data communication between memory and processing unit. We have thus chosen to study this kind of architecture and in particular, to check how some parameters of the architecture or of the devices influence the learning capabilities of the circuit.

3 Spiking Neural Network Simulation with N2S3

To perform our study of the influence of architecture and device parameters on the learning capabilities of SNN memristor-based hardware, we need to simulate this kind of circuits. In this section we motivate the need for and describe our simulator, N2S3.

3.1 Requirements for a Spiking Neuromorphic Hardware Simulator

In their comprehensive introduction and literature review about SNN, Paugam-Moisy and Bohte [38] explore the computational

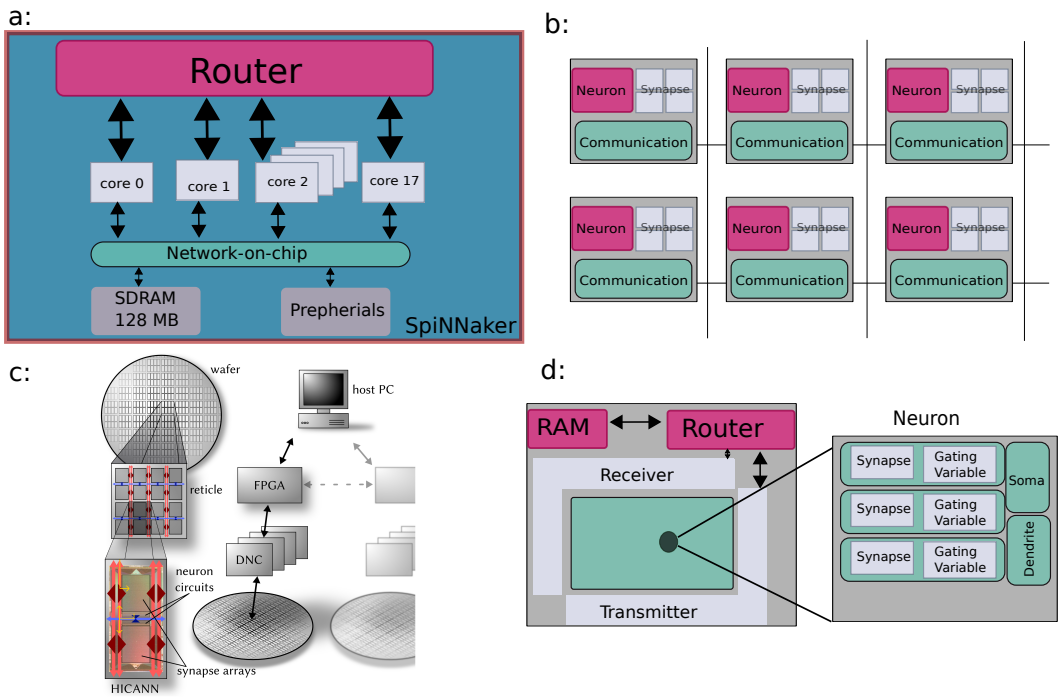

Fig. 2. Large scale spiking neural network systems, **a)** Principal architectural parts of a SpiNNaker processing node, **b)** In TrueNorth, conceptual blueprint of an architecture like the brain, tightly integrates memory, computation, and communication in distributed modules that operate in parallel and communicate via an event-driven. **c)** Schematic of HICANN board in BrainScales project, **d)** In Neurogrid, the chip comprises a 256×256 array of neuron elements, an asynchronous digital transmitter for sending the events generated by the neurons, a receiver block for accepting events from other sources, a router block for communicating packets among chips, and a memory blocks for supporting different network configurations.

capabilities of SNNs, their learning capabilities, and their simulation. Brette *et al.* [39] surveyed and discussed the existing work on SNN simulation in 2007. All the simulators discussed in this paper as well as the more recent Brian [23] simulator target the simulation of biological SNN. More recently, Bichler *et al.* [12] proposed Xnet, a C++ event-driven simulator dedicated to the simulation of hardware SNNs. In our work, we share the goals of Xnet (as stated in [12]): "intermediate modeling level, between low-level hardware description languages and high-level neural networks simulators used primarily in neurosciences", and "the integration of synaptic memristive device modeling, hardware constraints and any custom features required for the targeted application". In addition to these goals, we put an emphasis on *flexibility and usability* to allow the study of various kinds of hardware designs, *scalability* to simulate large hardware SNNs, and *software engineering best practices* (robust and extensible software architecture for maintainability, extensive test suite, continuous integration, open-source distribution).

In this section, we present N2S3 (Neural Network Scalable Spiking Simulator, pronounced "Nessie"), an event-driven simulator we have designed for the architecture exploration of hardware SNN architectures. The internals of N2S3 are based on the exchange of messages between concurrent actors [40], mimicking the exchange of spikes between neurons, thus allowing a lot of freedom in the network topologies. N2S3 has been developed from the ground up for extensibility, allowing to model various kinds of neuron and synapse models, various network topologies (especially, it is not restricted to feed-forward networks), various learning procedures (supervised or unsupervised), various reporting facilities, and to be user-friendly, with a domain specific language to easily express and run new experiments. It is available as open-source software at sourcesup.renater.fr/wiki/n2s3 so that its users can share their models and experimental settings to enable others to reproduce their results. In this spirit, N2S3 is distributed with the implementation of two "classical" experiments: handwritten digit recognition on the MNIST dataset [41], [42] and the highway vehicle counting experiment [43].

Thus the main differences with Xnet are the more flexible internal software architecture, the possibility to distribute a simulation on several computers, and the open source license (though Xnet has been released recently as a part of N2D2, an open source deep learning software platform, see github.com/CEA-LIST/N2D2). On the other hand Xnet can use GPUs to accelerate the computation while it is more complicated (and not done yet) with N2S3.

3.2 Event-Driven vs Clock-Driven Simulation

SNN are essentially defined by standard differential equations, but because of the discontinuities caused by the spikes, designing an efficient simulation of spiking neural networks is a non-trivial problem. There are two families of simulation algorithms: event-based simulators and clock-based ones. Synchronous or "clock-driven" simulation simultaneously updates all the neurons at every tick of a clock, and is easier to code, especially on GPUs, for getting an efficient execution of data-parallel learning algorithms. Event-driven simulation behaves more like hardware, in which conceptually concurrent components are activated by incoming signals (or events). As some of the learning mechanisms of spiking neural networks are based on the relative timings of spikes, the choice of the clock period for a clock-based simulation may lead either to imprecision or to a higher computational cost.

3.3 State of N2S3 as of version 1.0

To address our concurrency and distributability requirements (i.e., potential ability to scale out a simulation on several computers to handle large networks) we have chosen to use the Scala programming language [44] along with the Akka actor library [40]. We have demonstrated the feasibility to distribute a simulation but we have not yet tested the scalability of the simulations. This will be future work.

At the moment, N2S3 can read AER files and the MNIST files and convert them to spikes that are sent to an artificial neural network of any topology (we currently have shortcuts to model fully connected multilayer networks, convolutional networks, and reservoir computing topologies). The spikes are carried by messages between actors representing some subsets of the neurons of the network, and are fully or partially synchronized to offer a tradeoff between accuracy and concurrency. In the experiment below, messages are fully synchronized, and thus ordered by timestamps so that the accuracy of the simulation is guaranteed. The reader interested in more details about N2S3 can read [34] and the documentation at sourcesup.renater.fr/wiki/n2s3.

4 Experiment Setup and Models

We describe in this section the choices we have made to build our experiment: neuron and synapse models, network topology and learning process, and the dataset on which we run our simulations.

4.1 Leaky Integrate and Fire Neuron Model

The Leaky Integrate and Fire (LIF) neuron model is a well-studied model of neuron. There are three reasons for using LIF in our platform.

- The fabricated model with recent CMOS technology is available [45], [46].
- LIF works effectively in spiking and event-based networks [47].
- LIF models are quite fast to simulate, and particularly attractive for large-scale network simulations [39].

Neurons integrate the input spikes from other neurons they are connected to. These input spikes change the internal potential of the neuron, it is known as neuron's membrane potential or state variable. When this membrane potential passes a threshold voltage due to integrated inputs, the action potential occurs, in other words, the neuron fires. It then sends spikes to the output neurons it is connected to.

The model is described by the neuron membrane potential:

$$\tau_n \frac{\mathrm{d}v}{\mathrm{d}t} = -v(t) + RI_{syn}(t) \quad (1)$$

$$I_{syn}(t) = \sum_j g_{ij} \sum_n \alpha(t - t_j^{(n)}) \quad (2)$$

where $v(t)$ represents the membrane potential at time t, $\tau_n = RC$ is the membrane time constant and R is the membrane resistance. Equation 1 describes a simple parallel resistor-capacitor (RC) circuit where the leakage term is due to the resistor and the integration of $I_{syn}(t)$ is due to the capacitor. The total input current, $I_{syn}(t)$, is generated by the activity of pre-synaptic neurons. In fact, each pre-synaptic spike generates a post-synaptic current pulse. The total input current, injected to a neuron is the sum over all current pulses which is calculated in Equation 2. Time $t_j^{(n)}$ represents the time of the n_{th} spike of post-synaptic neuron j, and g_{ij} is the conductance of synaptic efficacy between neuron i and neuron j. Function $\alpha(t) = q\delta(t)$, where q is the injected charge to the artificial synapse and $\delta(t)$ is the Dirac pulse function. If $I_{syn}(t)$ is big enough where action potential can pass the threshold voltage, neuron fires. It means there are enough input spikes in a short time window. When there is no or only a few spikes in a time window, the neuron is in the leaky phase and the state variable decreases exponentially. The duration of this time window depends on $\tau_n = RC$. The equation is analytically solvable and thus we use the answer of Equation 1 in the network simulation when there is an input spike to improve the simulation performance. Figure 3.a shows the model of a single neuron. When the input voltage passes the threshold, the neuron fires and resets to its resting state. The membrane potential stays below the reset value for an definite period, which is called the *refractory* period.

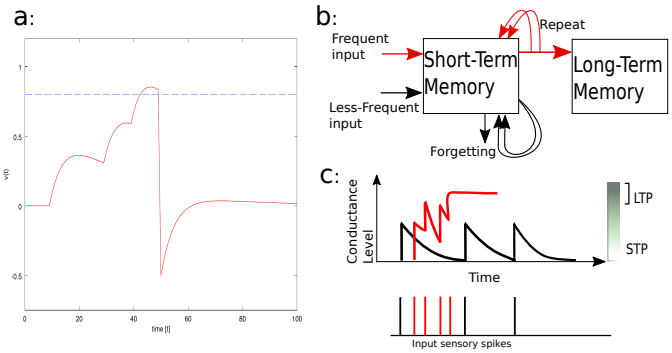

Fig. 3. **a)** Simulation of a single LIF neuron, the input spikes are applied in t=[10, 30, 40, 50] ms. Between 10 and 30 there is more decrease than between 30 and 40. **b)** Memorization in biology. **c)** The data is stored in Long-Term Memory (LTM) if the spikes are repeated in a certain time-window, otherwise Short-Term Memory (STM) will store temporary data.

4.2 Artificial Synapse Model

Before the discovery of a memristor nanodevice, by using state-of-the-art technology, 12 transistors were combined to mimic the behavior of memristor to perform the STDP learning method [48]. Therefore, using a two-terminal and scalable device such as the memristor could save a remarkable amount of power and cost specially in modeling large scale Spiking Neural Networks. To model biological synapses, not only do we need a device able to store the last activity, but it must also have enough flexibility to achieve Spike Timing Dependent Plasticity (STDP) for learning. Using memristor as a nonvolatile synaptic memory has been proposed in several works [16], [36], [37], [49]. By using nonvolatile memory, we can guarantee to store the last synaptic weight which is necessary for network training but the synapse can not forget. To be able to have a synapse which is able to forget, scientists used a volatile memory cell [50], [51]. We have proposed in [21] to combined both kinds of memories in order to improve the learning capabilities of the network. The results of that study show (Figure 4) that this combined synapse box (VNV synapse model on the figure) improves slightly the recognition rate on the MNIST benchmark, all other parameters identical.

The present study is complementary by looking at the effect of other parameters than the device used for the synapse on these learning capabilities. We use here the resistive RAM as modeled in our previous work [52]. By changing the doped-undoped regions

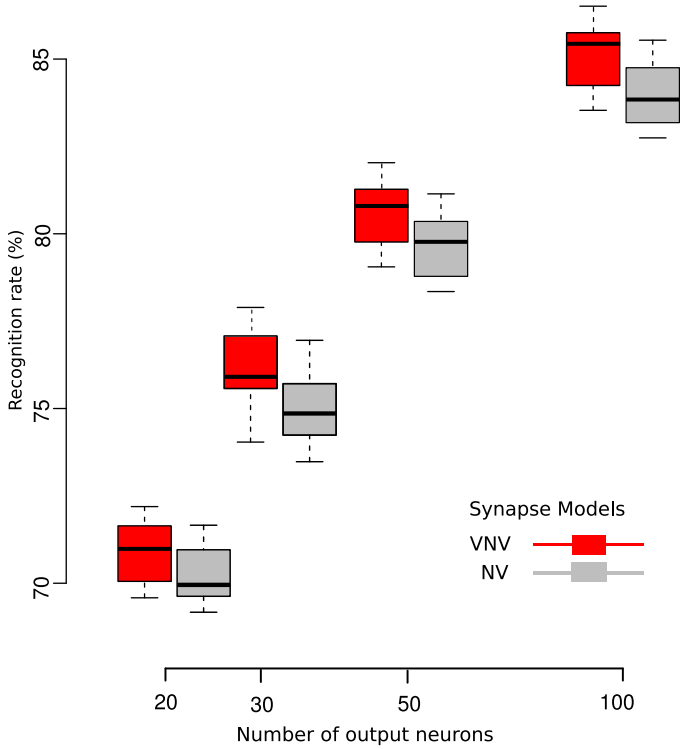

Fig. 4. Recognition rate as a function of number of output neurons. In the box plot (with the default parameters of the R boxplot function) for each number of neurons, we compare the recognition rate of the two synapse models.

of device, the conductance will be changed. A larger doped region leads to more conductivity. Therefore by controlling this boundary between two regions, the conductivity is controlled. The behavior of memristor can be modeled as follows [53]:

$$v(t) = R_m i(t) \tag{3}$$

$$R_m = R_{ON}\frac{w(t)}{D} + R_{OFF}\left(1 - \frac{w(t)}{D}\right) \tag{4}$$

where R_m is the variable resistance of memristor, $w(t)$ is the width of the doped region, D is the overall thickness of device, R_{ON} and R_{OFF} are device resistances while the active region is completely doped ($w = D$) and mostly undopped ($w \to 0$) respectively. To model the changing of the conductance, we use the model extracted from Equation 4 and introduced in [17], [54] by considering $g_{max} = \frac{1}{R_{ON}}$ and $g_{min} = \frac{1}{R_{OFF}}$ as the maximum and minimum device conductance respectively.

4.3 Network Topology and Learning Process

By using unsupervised learning inspired by biological neural networks, we propose a fully connected feed-forward network architecture. To figure out the correlation between the data, the STDP adjusts the strength of a synapse if the sensory input spikes are frequent enough to pass the short term potentiation and remain in the long term potentiation. In STDP, if there is an output spike in the pre-synaptic neuron and shortly after in the post-synaptic neuron, the conductance of the synapse between these two neurons increases. On the other hand, if the post-synaptic neuron spikes shortly before the pre-synaptic neuron, the conductance of the synapse between the two neurons decreases. A more comprehensive explanation for STDP is beyond the scope of this research, however if readers want to know how plasticity in memristor helps targeting STDP achievement we refer you to [55].

Furthermore, by inspiring from the biological behavior of the brain, we apply lateral inhibition inside a layer to reduce the activity of the neighbors of the winner neurons. This method is known as the winner-take-all (WTA) strategy [56]. The neuron which reaches the threshold first sends an inhibitory signal to all other neurons in the same layer to reset their states during the inhibition time.

The last issue in network architecture that we should address is *homeostasis*. In STDP learning, the connectivity between two neurons (i.e. the synaptic weight or conductance) is increased when the post synaptic neuron fires shortly after the presynaptic neuron. This process may be repeated frequently specially with WTA lateral inhibition. Homeostasis is a neuron property that regulates the firing threshold to prevent a neuron to be hyperactive [57]. The idea is to use an adaptive threshold for the membrane potential. If the neuron is too active in a short time window the threshold grows gradually; likewise, when a neuron is not active in a certain time window the threshold is reduced slightly.

4.4 MNIST Handwritten Digit Recognition Setup

We have used the MNIST training dataset of handwritten digits [22] to train and test the performance of neural networks based on the synapse box. The training set consists of 60000 digits between 0 and 9 and each handwritten number is a 28×28 pixel 8 bit gray scale image. In this simulation, we present the full dataset (60000 images) and full images twice. Each pixel is connected to one input buffer neuron. To transfer the image of a handwritten digit to spikes train, we have tried several spike distributions such as Poisson, Uniform, and Jitter. We have chosen the Poisson distribution, however we have not observed remarkable difference regarding to the output results of network efficiency (see section 5.1). Pixel intensity is between 0 to 255 and is transferred to 0 to 22 Hz spiking frequency using various distributions during a 350 ms presentation window. Based on previous similar work [58], we have chosen a delay of 150 ms between two images. Therefore, there is sufficient time for membrane potentials of all neurons to reset back to their initial value. The network connection weights are between 0 and 1 initialized using a Gaussian distribution.

The hardware platform is a 4 core Intel core i7 CPU. We have simulated different network topologies consisting of 2 fully interconnected layers, with a fixed number of input neurons ($28 \times 28 = 784$) and different numbers of output neurons (from 20 to 100 neurons).

To measure and evaluate the network classification accuracy after a fully unsupervised learning period consisting of the presentation of the full MNIST data set, we label the output neurons using 10000 samples of MNIST. After training, we stop all synaptic modification processes such as STDP. We assign a number class to the output neuron which has most frequent firing rate during the presentation of the 10000 labeling samples. Using these labels, we then evaluate the recognition rate of the network on 10000 different test samples by comparing the known class of the input with the class of the most frequent firing output neuron.

We obtain recognition rates that are comparable to the state of the art [17], [58] and the running time of the simulations are also comparable to those of similar experiments though it is difficult to make accurate comparisons because we lack precise information on the hardware and software setup of those works.

5 Experimental Evaluation of the Influence of Four Parameters on the Classification of Handwritten Digits

Not only choosing the network elements such as type of neuron (e.g., LIF), model of synapse, network architecture (e.g., RBM in this work) and algorithms of learning (e.g., STDP in this work) is important in the network recognition accuracy, but also tuning the parameters of each elements could have remarkable impact of recognition performance of network. For instance, evaluation of the impact of the number of neurons on recognition rate of network has been represented in [59], [60]. However this evaluation is for artificial neural network not spiking architecture. Another example is to evaluate the impact of learning parameter (α) and initial conductance of synapse which are studied in [17]. The advantages and disadvantages of using different types of STDP learning methods are studied comprehensively in [20], [61].

In this section, we evaluate and explore the effects of four parameters on the performance of the network architecture namely the distribution of input spike trains, the STDP window duration, the neuron threshold, and the synapse β parameter. Furthermore, due to importance of the number of neurons in hardware implementation of Neuromorphic architectures, we analyze these four parameters each time using different number of neurons. The results of these evaluations are useful for tuning the parameters for any SNN architecture using threshold-based model of neuron such as LIF, artificial model of synapse, and STDP learning method. Although the memristor is used to model the synapse in the network, however more case-specification to target more audiences in neuromorphic domain, we represent a general model of memristor which is close to biological model and represented in [17], [54]. The MNIST use-case is used in this evaluation, however the spiking distribution evaluation could be valid for other experimental use-cases as transferring data to spikes is common process for all SNN implementations.

These parameters will be fully described in the following. The full MNIST dataset is presented twice to the networks with four different numbers of output neurons. A sample of output results is shown in Figure 5 for 20, 30, 50 and 100 output neurons.

We vary each parameter independently to asses its influence on the recognition rate. The default value for these parameters are taken from the values listed in the literature. We call this set of parameters the baseline.

- Input spike train distribution: Poisson.
- STDP window duration: 25 ms.
- Neuron threshold: 15 mV.
- Synapse β parameter: 1.5.

At the end of this section, in Section 5.5, we compare the baseline with the best parameters obtained separately and discuss the results.

In all the simulations, as the weight initialization is random, each simulation has been repeated 10 times and the results are shown in box plots with the default parameters of the box plot function of the R statistical tool.

5.1 Effect of Spike Distribution

As the computation in SNNs is done using spikes, we need to generate spike trains of the images of the MNIST dataset to extract a spike-based dataset from a frame-based dataset respecting the intensity of each pixel of images. The pixel intensity is between 0

Fig. 5. Sample heat maps of the synaptic weights after network training with four different numbers of output neurons (20, 30, 50 and 100). As it is obvious in the figure, for network using 20 output neurons there is ambiguity in recognition of digits 3, 5 and 8 (underfitting). For networks using more output neurons, we add the number of synapses and network has more chance of learning. In addition, in MNIST testing, we can add the number of neurons to 300 (it is not illustrated in this figure) while still increasing the network prediction ability. However there is limitation of this increment due to overfitting. For instance the network recognition using 500 output neurons is less than network with 300 output neurons.

to 255 and should be transferred to the spike trains to be readable by a SNN. For instance, Diehl and Cook [58] use a method where the maximum pixel intensity of 255 is divided by 4, resulting in input firing rates between 0 and 63.75 Hz. Yet there remains the question of which time interval between two spikes do we use to generate the spikes? Therefore, statistical distribution rules can help to generate appropriate spike trains when encoding the pixels.

5.1.1 Poisson Input Spike Train Distribution

Here we explain a particular class of random process called a Poisson spike train process. Let us define $\rho(t)$ response function to the input stimuli by

$$\rho(t) = \sum_{i=1}^{k} \delta(t - t_i) \tag{5}$$

where k is the total number of spikes in the spike train, and t_i defines the moment each spike occurs. The unit impulse signal is defined as

$$\delta(t) = \begin{cases} 1 & \text{if } t = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

The instantaneous firing rate (e.g., of a sensory neuron) can now be formally defined to be the expectation of the sensory input response function which is $r(t) = \langle \rho(t) \rangle$. The average spike count between times t_1 and t_2 can then be defined from the instantaneous firing rate

$$\langle n \rangle = \int_{t_1}^{t_2} r(t)dt, \tag{7}$$

the probability of a spike occurring during a given brief time interval is equal to the value of the instantaneous firing rate during that time interval times the length of the interval.

$$P_{(\text{one spike in } (t-\delta t, t+\delta t))} = r(t)\delta t \tag{8}$$

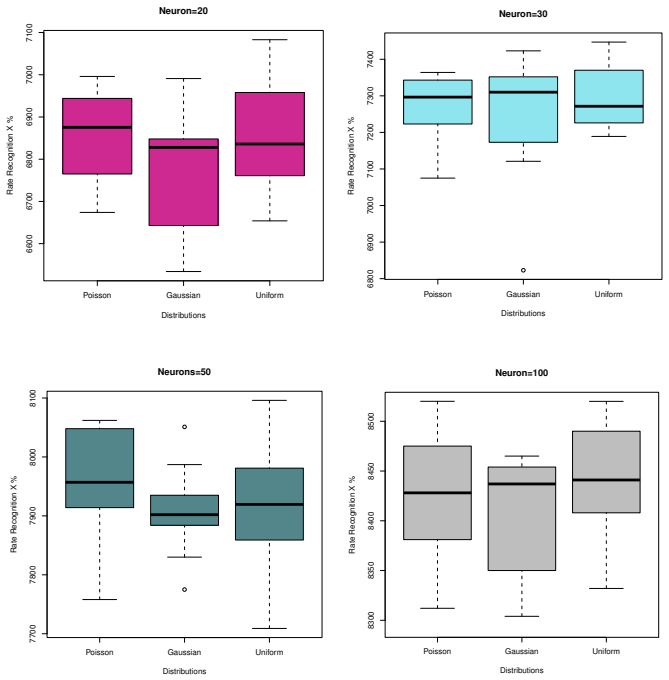

Fig. 6. Comparison of three different distributions for generating the input spike trains.

We assume the instantaneous firing rate r is constant over time. This is called a homogeneous Poisson process.

From Equation 7 we derive $\langle n \rangle = r\Delta t$ for any interval $\Delta t = t_2 - t_1$. Equation 8 can be used to generate a Poisson spike train by first subdividing time into a series of short intervals, each of duration δt and generating a sequence of random numbers $x[i]$ uniformly distributed between 0 and 1. For each interval, if $x[i] \leq r\delta t$, generate a spike, otherwise, no spike. This procedure is appropriate only when δt is small enough for e.g., millisecond range. Using Poisson distribution, we made a event-based version of MNIST [62] and it is available open-source online (github.com/MazdakFatahi/evt-MNIST). We refer to intensity of pixels as the probability that a spike occurs within an interval.

5.1.2 Other Distributions of the Input Spike Trains

In our simulation, the pixel intensity between 0 and 255 is transferred to 0 to 22 Hz spiking frequency using different probability distributions during a 350 ms presentation window. Based on previous similar work [58], we have chosen a delay of 150 ms between two images. Therefore, there is enough time for membrane potentials of all neurons to reset back to their initial value. The network connection weights are between 0 and 1 initialized using a Gaussian distribution. We compare uniform, Gaussian and Poisson distribution using different numbers of output neurons in Figure 6. This figure shows that the choice of the input distribution does not have a significant impact on the recognition rate of the network. That means that the classification of the inputs is done mainly on the frequency of the inputs regardless of the precise timing of the spikes.

Figure 7 depicts the average recognition rate for different numbers of neurons using the three spike train distributions. This figure demonstrate that increasing the number of neurons increases the recognition rate. We expected such a behavior because increasing the number of neurons will increase the number of

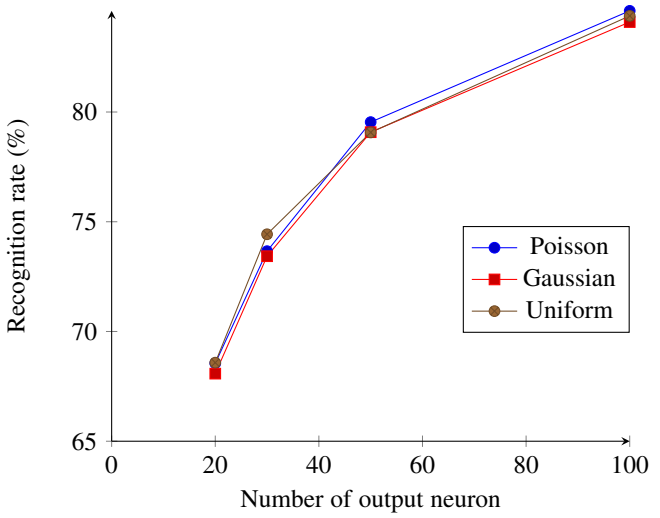

Fig. 7. Average recognition rate of the network using different input spike train distributions.

synapses in the network, therefore the chance of learning is increased due to more precise coding. It is worth noting that in general the relation between the number of neurons and the learning performance is not always increasing with the number of neurons. The reason is that the random selection of the number of neurons might cause either overfitting or underfitting problems.

5.2 Effect of STDP Window Duration

Activity-dependent modification of synaptic weights because of STDP depends on the correlations between pre- and postsynaptic firing over timescales of tens of milliseconds [63]. Contrarily to a biological simulation, as we simulate hardware SNNs, we can choose some parameters. The way STDP is implemented in hardware is a very important parameter. We use a simplified form of STDP (from [64]) where the weights of the synapses are always slightly decreased except when there is a temporal correlation between a presynaptic firing and a postsynaptic firing during a time window of duration STDP window. We look here at the influence of this STDP window duration on the learning capabilities of our networks.

The duration of the STDP window is related to the frequency of the input spike trains. We start using maximum frequency 63.75 Hz which is presented in [58], the STDP window duration should be of approximately the same duration as the corresponding period, 15.7 ms, or higher. To be able to evaluate the optimum STDP window duration, we have started using a 15 ms duration and increasing to 65 ms by increments of 10 ms as it is depicted in Figure 8.

The results show a low performance using 15 ms regarding to other STDP window durations. We have a remarkable improvement from 15 ms to 25 ms and the best results are obtained using the range between 35 ms and 55 ms which is reasonable corresponding maximum input spike frequency 22 Hz in our implementation. At 65 ms, the recognition rate starts to decrease. Our interpretation is that a too short duration does not allow to capture all the high intensity pixels in the input, and a too long duration is not specific enough as it captures too many mid range pixels and thus is not specific enough. One would need to do additional experiments to check how the STDP window influences the learning speed of the

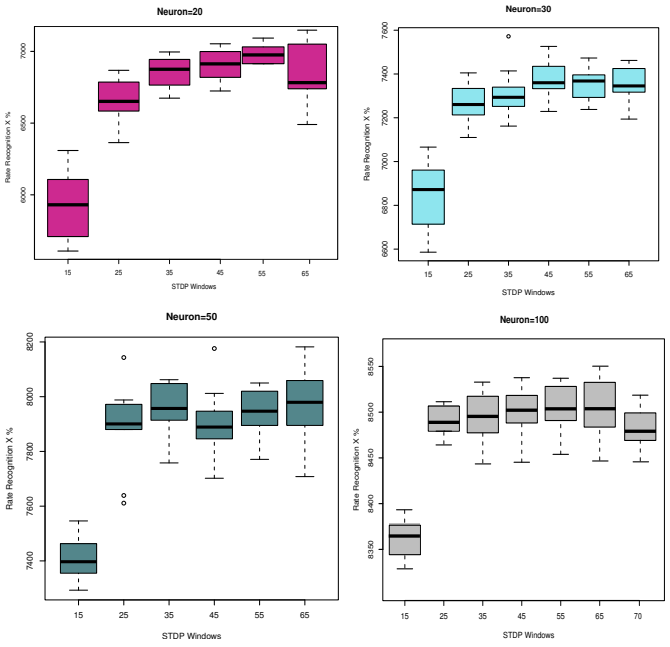

Fig. 8. The comparison of different STDP window durations.

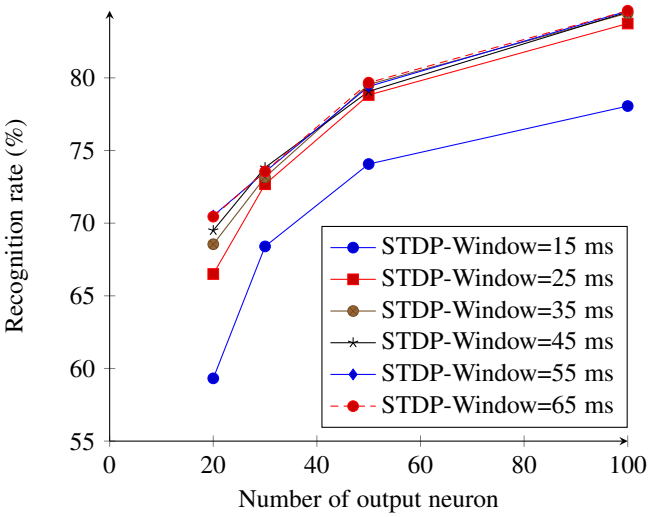

Fig. 9. Average recognition rate of the network using different number of neuron and six different STDP window durations.

network. Indeed, here we just check the final network state after two presentations of the 60000 images of the MNIST dataset, but the convergence speed of the SNN may well depend also on the STDP window duration.

Figure 9 illustrates the average recognition rate of neural networks using various number of neurons.

5.3 Effect of Neuron Threshold

In our hardware simulation, we use the Leaky-Integrate-and-Fire (LIF) neuron model. This type of neuron model is fit for SNN architectures [47] due to several properties such as availability of low-power CMOS design using subthreshold regime transistor [65], fast to simulate, and particularly efficient for large-scale network simulations [39].

The LIF model is described in Section 4.1, Equations 1 and 2. If injected currents from synapses are large enough, they cause the action potential to pass the threshold voltage, and the neuron fires. It means there are enough input spikes in a short time-window. When there is no or only a small number of spikes in a time-window, the neuron is in the leaky phase and the state variable decreases exponentially. The duration of this time window depends on $\tau_n = RC$.

To simulate the LIF neuron, we have to define a threshold for the neuron. Furthermore, we used the homeostasis technique to improve the network stability and performance. It means that if one or a group of neurons are too active when reading the input data, we slightly increase the threshold and vice versa if one or some neurons are inactive during the training process the threshold is decreased slightly. In this section, we verify and analyze the spiking neural network performance while using different thresholds for the same neurons to reach the best threshold value. To be able to trace only the impact of different thresholds on the system performance, we use the same homeostasis for all neurons with the same rate of increasing and reduction. The range of threshold values have been chosen to cover the minimum and maximum action potential in the network. Figures 10 and 11 shows the effect of the different thresholds on the recognition rate. The results show that the neuron thresholds between 25 and 35 mV lead to the best recognition rates. However, for the larger numbers of neurons choosing a threshold of 45 mV also leads to an acceptable performance. In contrast, for the networks with the smaller numbers of neurons the lower thresholds lead to a good recognition rate in the network. One could think that it is easily explained because in the networks with more neurons, each neuron is connected to a larger number of other neurons and thus receives more incoming spikes and thus reach its threshold sooner. But here, the number of inputs of the neurons is constant and equal to $28 \times 28 = 784$, the number of pixels in the image. Actually, the winner-takes-all rule increases the connectivity of the neurons inside the layer, but for inhibitory purposes. In general, the optimal value of the neuron threshold depends on the network architecture, and more generally on the activity of the network. In our case, the differences are not large enough to conclude and we would need to run a much larger number of simulations to check if this shift on the optimal value is real or just a random effect.

5.4 Effect of Synapse β Parameter

The quality of the synapse and the rate of conductance change of synapses causes influence the learning capabilities of SNNs. For instance using binary synapses which switch from zero to one will cause much lower performance in learning than using an analog synapse. Therefore the amount of change after updating the weights is important. The conductance change does not only depend on the STDP rule for modification but also on the characteristics of the synapse itself. Here, we evaluate the network learning performance when changing parameters in the synapse to modulate the synaptic weight change rate.

For the model of memristor as an artificial synapse, we use the model introduced in [17], [54] which is inspired from experimental memristive devices measurements [66]. The increasing and decreasing of conductance is presented by Equation 9 and Equation 10 respectively.

$$\Delta g_{\text{inc}} = \alpha_{\text{inc}} e^{-\beta \frac{g - g_{\min}}{g_{\max} - g_{\min}}} \tag{9}$$

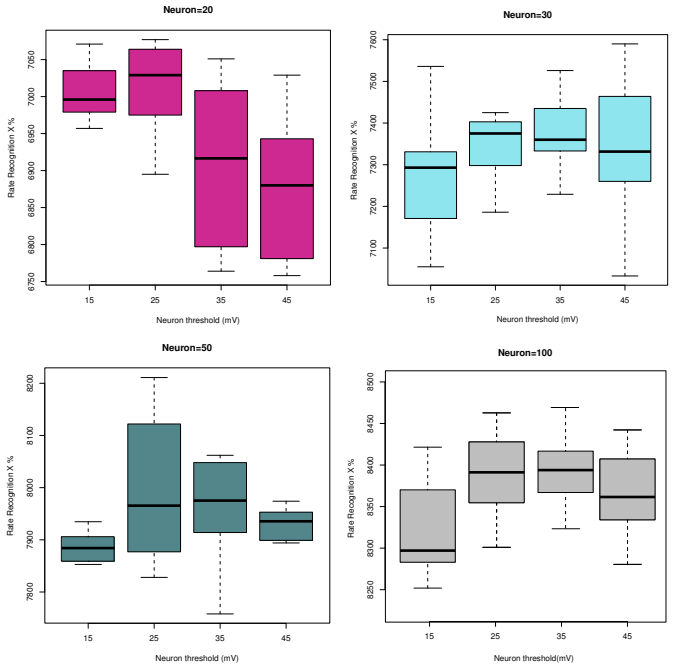

Fig. 10. Comparison of the recognition rate for various neuron thresholds.

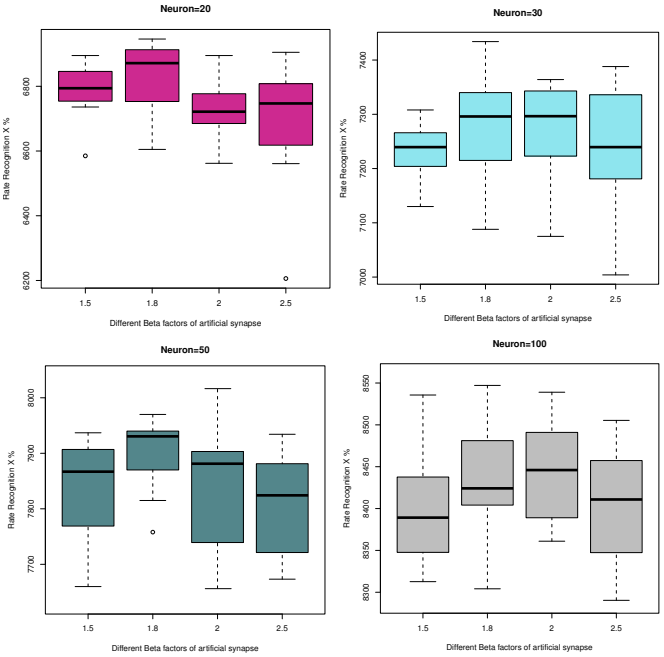

Fig. 12. Comparison of various fitting parameter (β) values. The results demonstrate better performance using β between 1.8 and 2. However, the differences are small.

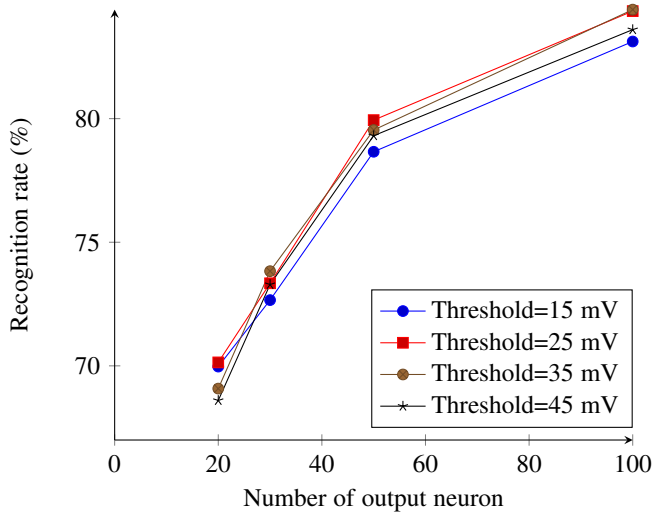

Fig. 11. Average recognition rate for various neuron thresholds.

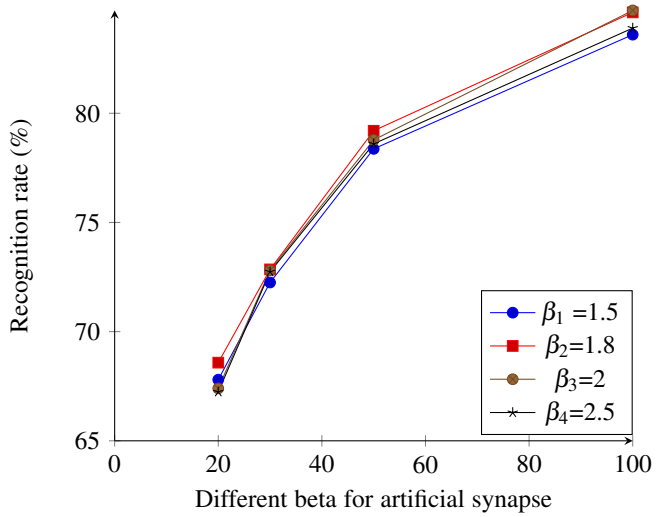

Fig. 13. comparing network performance using various number of neurons with different fitting parameter (β) for synapse model.

$$\Delta g_{\text{dec}} = \alpha_{\text{dec}} e^{-\beta \frac{g_{\max} - g}{g_{\max} - g_{\min}}} \tag{10}$$

$g_{\max}$ and $g_{\min}$ are the maximum and minimum of the conductance of the memristor. α_{inc} and α_{dec} characterize the conductance step and β is a fitting parameter. To evaluate the impact of synaptic conductance on learning in the neural network, we vary the β fitting parameter because it directly affects the amplitude of the weight modifications. The other parameters in the simulation are fixed to $g_{\max} = 1$, $g_{\min} = 0.0001$, $\alpha_{\text{inc}} = 0.01$ and $\alpha_{\text{dec}} = 0.005$. The initial conductance of the nanodevices before starting to train the network are chosen randomly around the mid-range. We observed no remarkable effective impact on the network performance, which is a proof of neural network robustness to variations as is reported in [64]. The results for four different number of neurons in the output are presented in Figure 12 and 13.

Although the network performance variation using different β parameters is not remarkable, the results demonstrate slightly better performance using β between 1.8 and 2.

5.5 Discussion

As a final experiment, the network architectures consist of non-volatile (values inspired by the literature [17], [64], [67] here NV) and volatile/nonvolatile (VNV) synapse models which is represented in [21] using the baseline parameters are compared with the best values for all the evaluated parameters, once including the nonvolatile (NV) synapse and once volatile/nonvolatile (VNV) synapse. The parameters are listed in table 1. Other network

TABLE 1
Best parameters vs. baseline parameters

	Synapse	Distrib.	STDP w.	Thres.	β
Best	VNV	Poisson	55 ms	35 mV	2
Best	NV	Poisson	55 ms	35 mV	2
Baseline	VNV	Poisson	25 ms	15 mV	1.5
Baseline	NV	Poisson	25 ms	15 mV	1.5

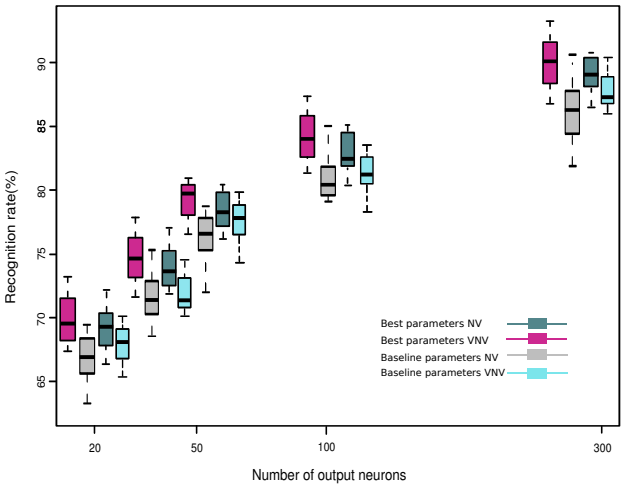

Fig. 14. Using the best parameters significantly improves the recognition rate.

elements and parameters that are not listed in table 1 are the same for all models.

We have evaluated these models using the same number of output neurons that were used in the parameter evaluation sections with the addition of 300 neurons for the output layer. The results in Figure 14 demonstrate enhanced performance using the best obtained parameters rather than baseline parameters. These results, as most simulations shown in this paper, show that the variation of recognition rate due to the random initialization of the synaptic weights is around 6 to 8 points with half of the results in a 3 to 4 point range. The ranges of variation of the best version and of the baseline overlap, so one does not have the guarantee that using theoretically better parameters will lead to a better performing network, it is just statistically better.

As a summary, we can conclude from this study that:

- The parameters that have a significant influence on the learning rate are the number of neurons in the SNN, the type of synapse (volatile or synapse box), the duration of the STDP window and the neuron threshold value.
- The distribution of the spikes in the input spike trains and the β fitting parameter of the volatile synapse do not have a significant impact on the recognition rate.
- One can significantly improve the recognition rate of a SNN by choosing better parameters.
- There is a relatively large spread of the recognition rate due to the random initialization of the synaptic weights in the simulations.

Thus a design space exploration should concentrate on the parameters that influence significantly the recognition rate, first of all the architecture of the network (number of neurons, topology), and then tune the other parameters. This design space exploration should be based on a statistical analysis of the performance of the network by running several simulations for each combination of parameter values. Finally, researchers should explain clearly what they mean when they give a single number as the recognition rate of a given SNN, is it a mean of several rates, the best rate or just a single rate.

6 Conclusion

We have presented an empirical study of the influence of several parameters on the recognition rate of memristor based hardware SNNs on the MNIST benchmark. This study is based on simulations run with N2S3, an open source simulator we have developed to help design neuromorphic circuits.

This study has shown that not all parameters have a significant influence on the learning rate of SNNs and thus that a design space exploration should concentrate first on the architecture of the network; then, the kind of device used as a synapse, the STDP mechanism used and its parameters, and the threshold of the neuron. Quantitatively we have found that the most significant parameters on our test case are the number of neurons (from less than 70 % of recognition rate with 20 neurons to around 85 % with 100 neurons), the model of synapse (difference of a few points of recognition rate in average between the two tested models) and its properties and the neuron threshold (a few points of improvement but rather a range of values that work similarly well). These numbers are averages and the reader should be well aware that due to the random initialization of the weights of the synapses the spread of the recognition rate for a given experiment (i.e. with fixed parameters) is of 4 to 5 points, so there is not a single combination of parameters that is always better than the others, only statistically better.

This study is only valid on the MNIST benchmark and should be complemented by similar studies on other test cases to confirm these findings, especially test cases using natural data in spiking form where the precise relative timings of the input spikes would necessitate more precise STDP mechanisms than the simplified one used in this paper that is only sensitive to the average frequency of the input spike trains.

In the future, we will explore different models of synapse and neurons, more complex network topologies and STDP mechanisms, and enhance the N2S3 simulator with automatic design space exploration facilities that will concentrate on optimizing the most significant parameters discovered in this study. In addition to the recognition rate (or classification capabilities), we will also evaluate other performance measures such as the power consumption of the circuit or its convergence speed. As there are various number of memristors, evaluation the proper one for neuromorphic architecture could be another potential study for future works specially with considering real fabrication parameters.

Acknowledgments

This work has been supported by IRCICA (Univ. Lille, CNRS, USR 3380 – IRCICA, F-59000 Lille, France).

REFERENCES

[1] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014. [Online]. Available: http://science.sciencemag.org/content/345/6197/668.abstract

[2] S. B. Eryilmaz, S. Joshi, E. Neftci, W. Wan, G. Cauwenberghs, and H.-S. P. Wong, "Neuromorphic architectures with electronic synapses." IEEE, Mar. 2016, pp. 118–123. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7479186

[3] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006. [Online]. Available: http://dx.doi.org/10.1162/neco.2006.18.7.1527

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. [Online]. Available: http://www.nature.com/nature/journal/v521/n7553/full/nature14539.html

[5] J. Yang, M. Pickett, X. Li, D. Ohlberg, D. Stewart, and R. Williams, "Memristive switching mechanism for metal/oxide/metal nanodevices," *Nature Nanotechnology*, vol. 3, no. 7, pp. 429–433, 2008. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-46749093701&partnerID=40&md5=f2a7152ab8e0922c0eabc0c44f89ee7b

[6] C. Xu, X. Dong, N. P. Jouppi, and Y. Xie, "Design implications of memristor-based rram cross-point structures." in *DATE*. IEEE, 2011, pp. 734–739.

[7] M. Prezioso, F. Merrikh-Bayat, B. Hoskins, G. Adam, K. K. Likharev, and D. B. Strukov, "Training and Operation of an Integrated Neuromorphic Network Based on Metal-Oxide Memristors," *Nature*, vol. 521, no. 7550, pp. 61–64, May 2015, arXiv: 1412.0611. [Online]. Available: http://arxiv.org/abs/1412.0611

[8] T. Driscoll, H.-T. Kim, B.-G. Chae, M. D. Ventra, and D. N. Basov, "Phase-transition driven memristive system," *Applied Physics Letters*, vol. 95, no. 4, p. 043503, Jul. 2009. [Online]. Available: http://scitation.aip.org/content/aip/journal/apl/95/4/10.1063/1.3187531

[9] S. B. Eryilmaz, D. Kuzum, R. Jeyasingh, S. Kim, M. BrightSky, C. Lam, and H.-S. P. Wong, "Brain-like associative learning using a nanoscale non-volatile phase change synaptic device array," *Neuromorphic Engineering*, vol. 8, p. 205, 2014. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fnins.2014.00205/full

[10] S. Ambrogio, N. Ciocchini, M. Laudato, V. Milo, A. Pirovano, P. Fantini, and D. Ielmini, "Unsupervised Learning by Spike Timing Dependent Plasticity in Phase Change Memory (PCM) Synapses," *Neuromorphic Engineering*, p. 56, 2016. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fnins.2016.00056/full

[11] Q. Liu, S. Long, H. Lv, W. Wang, J. Niu, Z. Huo, J. Chen, and M. Liu, "Controllable Growth of Nanoscale Conductive Filaments in Solid-Electrolyte-Based ReRAM by Using a Metal Nanocrystal Covered Bottom Electrode," *ACS Nano*, vol. 4, no. 10, pp. 6162–6168, Oct. 2010. [Online]. Available: http://dx.doi.org/10.1021/nn1017582

[12] O. Bichler, D. Roclin, C. Gamrat, and D. Querlioz, "Design exploration methodology for memristor-based spiking neuromorphic architectures with the Xnet event-driven simulator," in *2013 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, Jul. 2013, pp. 7–12.

[13] M. Suri, D. Querlioz, O. Bichler, G. Palma, E. Vianello, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Bio-Inspired Stochastic Computing Using Binary CBRAM Synapses," *IEEE Transactions on Electron Devices*, vol. 60, no. 7, pp. 2402–2409, Jul. 2013. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6523102

[14] 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," Jun. 2012. [Online]. Available: http://arxiv.org/abs/1206.3397

[15] Y. Nishitani, Y. Kaneko, and M. Ueda, "Artificial synapses using ferroelectric memristors embedded with CMOS Circuit for image recognition," in *72nd Device Research Conference*, Jun. 2014, pp. 297–298.

[16] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale Memristor Device as Synapse in Neuromorphic Systems," *Nano Letters*, vol. 10, no. 4, pp. 1297–1301, Apr. 2010. [Online]. Available: http://dx.doi.org/10.1021/nl904092h

[17] D. Querlioz, O. Bichler, P. Dollfus, and C. Gamrat, "Immunity to Device Variations in a Spiking Neural Network With Memristive Nanodevices," *IEEE Transactions on Nanotechnology*, vol. 12, no. 3, pp. 288–295, May 2013.

[18] W. Gerstner, R. Ritz, and J. L. Hemmen, "Why spikes? hebbian learning and retrieval of time-resolved excitation patterns," *Biological Cybernetics*, vol. 69, no. 5-6, pp. 503–515, 1993.

[19] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann, "Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps," *Science*, vol. 275, no. 5297, pp. 213–215, 1997. [Online]. Available: http://www.sciencemag.org/content/275/5297/213.abstract

[20] B. Babadi and L. F. Abbott, "Stability and Competition in Multi-spike Models of Spike-Timing Dependent Plasticity," *PLOS Comput Biol*, vol. 12, no. 3, p. e1004750, Mar. 2016. [Online]. Available: http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004750

[21] M. Shahsavari, P. Falez, and P. Boulet, "Combining a Volatile and Nonvolatile Memristor in Artificial Synapse to Improve Learning in Spiking Neural Networks," in *12th ACM/IEEE International Symposium on Nanoscale Architectures (Nanoarch 2016)*, Beijing, China, Jul. 2016.

[22] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, pp. 2278–2324.

[23] D. F. M. Goodman, R. Brette, D. Goodman, and R. Brette, "Brian: a simulator for spiking neural networks in Python," *Frontiers in Neuroinformatics*, vol. 2, p. 5, 2008. [Online]. Available: http://journal.frontiersin.org/article/10.3389/neuro.11.005.2008/full

[24] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker Project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, May 2014.

[25] 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," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, May 2014.

[26] J. Schemmel, D. Briiderle, A. Griibl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling." IEEE, May 2010, pp. 1947–1950. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5536970

[27] K. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 5, pp. 416–434, May 2000. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=842110

[28] E. Stromatias, D. Neil, F. Galluppi, M. Pfeiffer, S. C. Liu, and S. Furber, "Scalable energy-efficient, low-latency implementations of trained spiking Deep Belief Networks on SpiNNaker," in *2015 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2015, pp. 1–8.

[29] S. K. Esser, A. Andreopoulos, R. Appuswamy, P. Datta, D. Barch, A. Amir, J. Arthur, A. Cassidy, M. Flickner, P. Merolla, S. Chandra, N. Basilico, S. Carpin, T. Zimmerman, F. Zee, R. Alvarez-Icaza, J. A. Kusnitz, T. M. Wong, W. P. Risk, E. McQuinn, T. K. Nayak, R. Singh, and D. S. Modha, "Cognitive computing systems: Algorithms and applications for networks of neurosynaptic cores," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, Aug. 2013, pp. 1–10.

[30] K. Wendt, M. Ehrlich, and R. Schüffny, "A Graph Theoretical Approach for a Multistep Mapping Software for the FACETS Project," in *Proceedings of the 2Nd WSEAS International Conference on Computer Engineering and Applications*, ser. CEA'08. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2008, pp. 189–194. [Online]. Available: http://dl.acm.org/citation.cfm?id=1373936.1373969

[31] G. Indiveri and S.-C. Liu, "Memory and information processing in neuromorphic systems," *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1379–1397, Aug. 2015, arXiv: 1506.03264. [Online]. Available: http://arxiv.org/abs/1506.03264

[32] A. P. Davison, D. Brüderle, J. M. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger, "PyNN: a common interface for neuronal network simulators," *Frontiers in Neuroinformatics*, vol. 2, 2009. [Online]. Available: http://journal.frontiersin.org/article/10.3389/neuro.11.011.2008/full

[33] M. Shahsavari, P. Devienne, and P. Boulet, "N2s3, a Simulator for the Architecture Exploration of Neuromorphic Accelerators," Mar. 2015, neuComp in DATE. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01240444

[34] P. Boulet, P. Devienne, P. Falez, G. Polito, M. Shahsavari, and P. Tirilly, "N2s3, an Open-Source Scalable Spiking Neuromorphic Hardware Simulator," Université de Lille 1, Sciences et Technologies ; CRIStAL UMR 9189, Research Report, Jan. 2017. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01432133

[35] J. Borghetti, Z. Li, J. Straznicky, X. Li, D. A. A. Ohlberg, W. Wu, D. R. Stewart, and R. S. Williams, "A hybrid nanomemristor/transistor

logic circuit capable of self-programming," *Proceedings of the National Academy of Sciences*, vol. 106, no. 6, pp. 1699–1703, 2009.

[36] F. Alibart, S. Pleutin, D. Guérin, C. Novembre, S. Lenfant, K. Lmimouni, C. Gamrat, and D. Vuillaume, "An Organic Nanoparticle Transistor Behaving as a Biological Spiking Synapse," *Advanced Functional Materials*, vol. 20, no. 2, pp. 330–337, Jan. 2010. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/adfm.200901335/abstract

[37] G. Indiveri, B. Linares-Barranco, R. Legenstein, G. Deligeorgis, and T. Prodromakis, "Integration of nanoscale memristor synapses in neuromorphic computing architectures," *Nanotechnology*, vol. 24, no. 38, p. 384010, Sep. 2013, arXiv: 1302.7007. [Online]. Available: http://arxiv.org/abs/1302.7007

[38] H. Paugam-Moisy and S. Bohte, "Computing with Spiking Neuron Networks," in *Handbook of Natural Computing*, G. Rozenberg, T. Bäck, and J. N. Kok, Eds. Springer Berlin Heidelberg, 2012, pp. 335–376, dOI: 10.1007/978-3-540-92910-9_10. [Online]. Available: http://link.springer.com/referenceworkentry/10.1007/978-3-540-92910-9_10

[39] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, M. Diesmann, A. Morrison, P. H. Goodman, F. C. Harris, M. Zirpe, T. Natschläger, D. Pecevski, B. Ermentrout, M. Djurfeldt, A. Lansner, O. Rochel, T. Vieville, E. Muller, A. P. Davison, S. El Boustani, and A. Destexhe, "Simulation of networks of spiking neurons: a review of tools and strategies," *Journal of Computational Neuroscience*, vol. 23, no. 3, pp. 349–398, Dec. 2007.

[40] D. Wyatt, *Akka Concurrency*. USA: Artima Incorporation, 2013.

[41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[42] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995. [Online]. Available: https://www.researchgate.net/profile/Yann_Lecun/publication/2453996_Convolutional_Networks_for_Images_Speech_and_Time-Series/links/0deec519dfa2325502000000.pdf

[43] O. Bichler, D. Querlioz, S. J. Thorpe, J.-P. Bourgoin, and C. Gamrat, "Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity," *Neural Networks*, vol. 32, pp. 339–348, Aug. 2012.

[44] M. Odersky, P. Altherr, V. Cremet, I. Dragos, G. Dubochet, B. Emir, S. McDirmid, S. Micheloud, N. Mihaylov, M. Schinz, E. Stenman, L. Spoon, and M. Zenger, "An overview of the Scala programming language (2nd Edition)," École Polytechnique Fédérale de Lausanne (EPFL), Technical Report LAMP-REPORT-2006-001, 2006.

[45] S.-C. Liu and R. Douglas, "Temporal coding in a silicon network of integrate-and-fire neurons," *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1305–1314, Sep. 2004.

[46] E. Chicca, D. Badoni, V. Dante, M. D'Andreagiovanni, G. Salina, L. Carota, S. Fusi, and P. D. Giudice, "A VLSI recurrent network of integrate-and-fire neurons connected by plastic synapses with long-term memory," *IEEE Transactions on Neural Networks*, vol. 14, no. 5, pp. 1297–1307, Sep. 2003.

[47] W. Maass and C. M. Bishop, Eds., *Pulsed Neural Networks*. Cambridge, MA, USA: MIT Press, 1999.

[48] G. Indiveri, "Neuromorphic bistable VLSI synapses with spike-timing-dependent plasticity," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 15. Cambridge, MA, USA: MIT Press, December 2003, pp. 1091–1098. [Online]. Available: http://ncs.ethz.ch/pubs/pdf/Indiveri03c.pdf

[49] H. Kim, M. P. Sah, C. Yang, T. Roska, and L. O. Chua, "Memristor Bridge Synapses," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 2061–2070, Jun. 2012.

[50] F. Alibart, S. Pleutin, O. Bichler, C. Gamrat, T. Serrano-Gotarredona, B. Linares-Barranco, and D. Vuillaume, "A Memristive Nanoparticle/Organic Hybrid Synapstor for Neuroinspired Computing," *Advanced Functional Materials*, vol. 22, no. 3, pp. 609–616, Feb. 2012. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/adfm.201101935/abstract

[51] S. Desbief, A. Kyndiah, D. Guérin, D. Gentili, M. Murgia, S. Lenfant, F. Alibart, T. Cramer, F. Biscarini, and D. Vuillaume, "Low voltage and time constant organic synapse-transistor," *Organic Electronics*, vol. 21, pp. 47–53, Jun. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566119915000786

[52] M. Shahsavari, M. Faisal Nadeem, S. Arash Ostadzadeh, P. Devienne, and P. Boulet, "Unconventional digital computing approach: memristive nanodevice platform," *physica status solidi (c)*, vol. 12, no. 1-2, pp. 222–228, Jan. 2015. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/pssc.201400069/abstract

[53] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008. [Online]. Available: http://www.nature.com/nature/journal/v453/n7191/full/nature06932.html

[54] D. Querlioz, P. Dollfus, O. Bichler, and C. Gamrat, "Learning with memristive devices: How should we model their behavior?" in *2011 IEEE/ACM International Symposium on Nanoscale Architectures*, Jun. 2011, pp. 150–156.

[55] T. Serrano-Gotarredona, T. Masquelier, T. Prodromakis, G. Indiveri, and B. Linares-Barranco, "STDP and STDP variations with memristors for spiking neuromorphic learning systems," *Frontiers in Neuroscience*, vol. 7, p. 2, 2013. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fnins.2013.00002/full

[56] B. Nessler, M. Pfeiffer, and W. Maass, "STDP enables spiking neurons to detect hidden causes of their inputs," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 1357–1365. [Online]. Available: http://papers.nips.cc/paper/3744-stdp-enables-spiking-neurons-to-detect-hidden-causes-of-their-inputs.pdf

[57] E. Marder and J.-M. Goaillard, "Variability, compensation and homeostasis in neuron and network function," *Nature Reviews Neuroscience*, vol. 7, no. 7, pp. 563–574, Jul. 2006. [Online]. Available: http://www.nature.com/nrn/journal/v7/n7/full/nrn1949.html

[58] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in Computational Neuroscience*, p. 99, 2015. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fncom.2015.00099/full

[59] Y. Liu, J. A. Starzyk, and Z. Zhu, "Optimizing number of hidden neurons in neural networks," in *Proceedings of the 25th Conference on Proceedings of the 25th IASTED International Multi-Conference: Artificial Intelligence and Applications*, ser. AIAP'07. Anaheim, CA, USA: ACTA Press, 2007, pp. 121–126. [Online]. Available: http://dl.acm.org/citation.cfm?id=1295303.1295324

[60] S. Lawrence, C. L. Giles, and A. C. Tsoi, "What Size Neural Network Gives Optimal Generalization? Convergence Properties of Backpropagation." [Online]. Available: http://clgiles.ist.psu.edu/papers/UMD-CS-TR-3617.what.size.neural.net.to.use.pdf

[61] A. Morrison, M. Diesmann, and W. Gerstner, "Phenomenological models of synaptic plasticity based on spike timing," *Biological Cybernetics*, vol. 98, no. 6, pp. 459–478, Jun. 2008. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2799003/

[62] M. Fatahi, M. Ahmadi, M. Shahsavari, A. Ahmadi, and P. Devienne, "evt_mnist: A spike based version of traditional MNIST," *arXiv:1604.06751 [cs]*, Apr. 2016, arXiv: 1604.06751. [Online]. Available: http://arxiv.org/abs/1604.06751

[63] P. J. Drew and L. F. Abbott, "Extending the effects of spike-timing-dependent plasticity to behavioral timescales," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 23, pp. 8876–8881, Jun. 2006. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1470971/

[64] D. Querlioz, O. Bichler, and C. Gamrat, "Simulation of a memristor-based spiking neural network immune to device variations," in *The 2011 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2011, pp. 1775–1781.

[65] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. Saighi, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen, "Neuromorphic Silicon Neuron Circuits," *Frontiers in Neuroscience*, vol. 5, May 2011. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3130465/

[66] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale Memristor Device as Synapse in Neuromorphic Systems," *Nano Lett.*, vol. 10, no. 4, pp. 1297–1301, Mar. 2010.

[67] A. Sboev, D. Vlasov, A. Serenko, R. Rybka, and I. Moloshnikov, "A comparison of learning abilities of spiking networks with different spike timing-dependent plasticity forms," *Journal of Physics: Conference Series*, vol. 681, no. 1, p. 012013, 2016. [Online]. Available: http://stacks.iop.org/1742-6596/681/i=1/a=012013

Mahyar Shahsavari is a Lecturer (ATER) in computer science and a researcher in CRIStAL lab at Univ. Lille. He received BSc and MSc in Electronics at Razi University. He received a PhD in Computer Science from Lille University under supervision of Pierre Boulet in 2016.

His research interests are unconventional computing using emerging technologies (Memristor, CNTFET, etc), neuromorphic, cognitive computing, machine learning and neural network application in embedded system as well as big data analyzing.

Contact him at mahyar.shahsavari@gmail.com.

Pierre Boulet is a full professor of computer science at Univ. Lille, France. He received a PhD of computer science in 1996 from the École Normale Supérieure de Lyon, France.

His interests range from parallelism, compilation, embedded system co-design to model driven engineering and synchronous languages. He is currently investigating how to program time and energy aware mobile applications on post-Moore architectures, and help design neuromorphic accelerators.

He is deputy director of CRIStAL (computer science and automatics laboratory of Lille). He is a member of the HiPEAC EU support action, and of the IEEE and ACM professional societies. Contact him at Pierre.Boulet@univ-lille.fr.