Bioinspired Programming of Memory Devices for Implementing an Inference Engine

Damien Querlioz, Olivier Bichler, Adrien Francis Vincent, Christian Gamrat

Bioinspired Programming of Memory Devices for Implementing an Inference Engine

Damien Querlioz, *Member, IEEE,* Olivier Bichler, Adrien F. Vincent *Student Member, IEEE,* and Christian Gamrat

***Abstract*—Cognitive tasks are essential for the modern applications of electronics, and rely on the capability to perform inference. The Von Neumann bottleneck is an important issue for such tasks, and emerging memory devices offer an opportunity to overcome this issue by fusing computing and memory, in nonvolatile instant ON / OFF systems. A vision for accomplishing this is to use brain-inspired architectures, which excel at inference and do not differentiate between computing and memory. In this work, we use a neuroscience inspired model of learning, spike timing dependent plasticity, to develop a bioinspired approach for programming memory devices, which naturally gives rise to an inference engine. The method is then adapted to different memory devices, including multivalued memories (cumulative memristive device, phase change memory) and stochastic binary memories (conductive bridge memory, spin transfer torque magnetic tunnel junction). By means of system-level simulations, we investigate several applications including image recognition, and pattern detection within video and auditory data. We compare the results of the different devices. Stochastic binary devices require the use of redundancy, the extent of which depends tremendously on the considered task. A theoretical analysis allows us to understand how the various devices differ, and ties the inference engine to the machine learning algorithm of Expectation-Maximization. Monte Carlo simulations demonstrate an exceptional robustness of the inference engine with respect to device variations and other issues. A theoretical analysis explains the roots of this robustness. These results highlight a possible new bioinspired paradigm for programming emerging memory devices, allowing the natural learning of a complex inference engine. The physics of the memory devices plays an active role. The results open the way for a reinvention of the role of memory, when solving inference tasks.**

***Index Terms*—memory devices, neural networks, inference.**

I. Introduction

MODERN electronics applications for smart sensors, medical electronics and the future Internet-of-Things require electronic circuits capable of handling large volumes of noisy and incomplete real-life data. Tasks like recognition or mining, broadly qualified as "cognitive", are necessary. While such assignments are not hard for humans, because of their natural capability to perform inference, performing them with computers necessitates advanced machine learning algorithms. In contrast to more traditional programming, these algorithms require minimal computing but significant memory access [1]. For this reason, cognitive algorithms are severely affected by the separation of computing and memory, known as the Von Neumann bottleneck, and need big computers with a large power budget. For example, a recent widely publicized demonstration by Google of a "deep" network uses no complicated equations but accesses one billion parameters from memory [2] (see also Figure 1). Due to the inefficiency of cognitive tasks running on computers, smart devices currently must rely on the cloud and its megawatt data centers.

Emerging nonvolatile memory devices (e.g. resistive memory [3], memristor [4], [5], magnetic memory [6]) may be a major advancement for this situation. As they are quite compact and can be embedded in CMOS circuitry, they offer an opportunity to fuse computing and memory, with the additional benefit of nonvolatility. Cognitive systems, the parameters of which would be stored in nonvolatile memory at the core of CMOS, would provide instant ON/OFF and offer the possibility of highly reduced power consumption for the smart objects of the Internet-of-Things. However, designing such systems requires a complete reinvention of electronics around cognitive tasks and inference problems, in stark contrast with the Von Neumann paradigm. A current idea is to try to use the human brain directly as an inspiration for a new type of architecture. Indeed, brains excel at inference and do not differentiate between computing and memory: memory is embedded at the core of computation.

Although new memory devices are particularly appealing for microelectronics, they also pose many challenges. They often suffer from high device variability, noise and partly non repeatable behavior [7], [8]; very much like the synapses and ion channels that the brain uses for computing [9]. Despite this, the brain achieves exceptional inference capability.

For these reasons, in recent years, the exploitation of emerging memory devices in brain inspired systems has stimulated a considerable interest. They are typically proposed to be used as synapses between silicon neurons [4], [10]–[29]. Such efforts fit particularly well with recent progress on the implementation of large neuroinspired systems [30]–[34], realizing a vision pioneered by the research of Carver Mead in the late 80's [35].

Building on these ideas, here we show an example of how simple stochastic programming of memory devices can lead to a system capable of performing complex inference. We differentiate between two main approaches: one that exploits multilevel memory effects, and one that relies on binary memory devices programmed in a stochastic way, possibly employing device redundancy. We provide many examples, exploring a range of inference tasks and device physics phenomena. We introduce a theoretical analysis for the systems functionality, based on [36]. We explain in detail how the device physics relates to the inference capability, and how this

D. Querlioz and A. F. Vincent are with Institut d'Electronique Fondamentale, Univ. Paris-Sud, CNRS, 91405 Orsay France (email: damien.querlioz@u-psud.fr; http://www.ief.u-psud.fr/~querlioz/).

O. Bichler and C. Gamrat are with CEA, LIST, Saclay, France.

Manuscript received XXX. This work was supported by the ANR COGNISPIN (ANR-13-JS03-0004-01) and the FP7 ICT BAMBI (FP7-ICT-2013-C) projects, and the CNRS/MI DEFI NANO program.

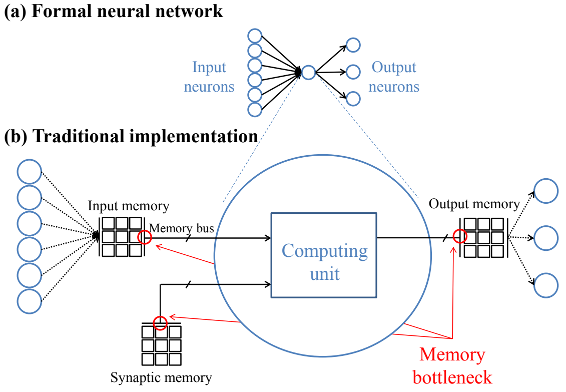

Fig. 1. Implementation of cognitive algorithms like neural network (illustrated in (a)) using a Von Neumann architecture suffers heavily from the separation of computing and memory – the Von Neumann bottleneck. A neuron only performs basic operations, which depend on a high number of memory access, as illustrated in (b).

in turn relates to sophisticated machine learning ideas and to Bayesian inference. Finally, we explore the intrinsic robustness of these approaches with regards to device issues.

II. Stochastic Programming of an Inference Engine

A. General Ideas

In brains, the capability for inference naturally emerges from learning [38]. Though the underlying mechanism is far from understood, some models of how learning occurs at the synaptic and neuronal level provide inspiration for novel electronics paradigms. It is widely accepted that long term memory is stored in synapses, which are the connections between the neurons, which are the active computation units of the brain. Synapses not only transmit information from one neuron to another, but also adjust their strength (usually called "synaptic weight") in response to experience. This adaptation, known as synaptic plasticity, is thought to be the most important phenomenon for long term learning.

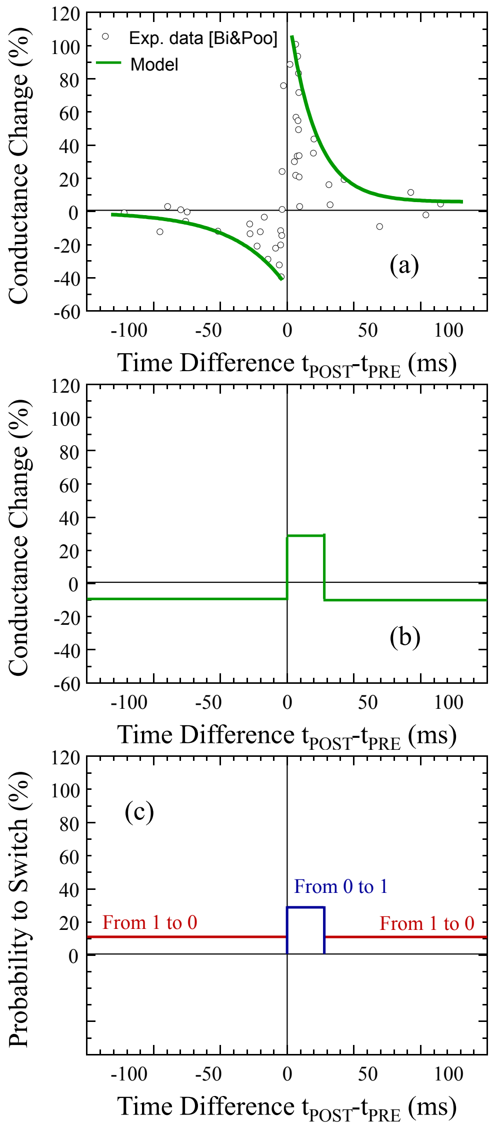

Fig. 2. Variations of the STDP curve. (a) Original measurement from biology [37] and the traditional model. (b) Simplified STDP used in our inference engine. (c) Stochastic version of the simplified STDP.

A canonical model of synaptic plasticity, known as spike timing dependent plasticity (STDP), was identified in the late 90's [37], [39]. It was proposed to provide an intuitive interpretation of experimental observations. Neurons communicate by asynchronous spikes, which are transmitted by synapses from "presynaptic neurons" to "postsynaptic neurons". The assumption underlying STDP is that synapses tend to reinforce causal links. That is, when the presynaptic neuron spikes just before the postsynaptic neuron spikes, the synapse between the two becomes stronger. Therefore, if the presynaptic neuron spikes again, the synapse will allow the postsynaptic neuron to spike faster. In contrast, when the postsynaptic neuron spikes just before the presynaptic neuron, the synapse becomes weaker. This is summarized in the conventional STDP curve of Figure 2(a). In addition to this conventional curve, many variations of STDP have been observed experimentally (e.g. symmetric, asymmetric, voltage-dependent...) [40]. One should be careful to note that neuroscientists consider STDP a simplified model; it does not account for all that has been observed regarding synaptic plasticity and should not be seen as the only learning process the brain possesses [41]. However, research in neuroscience has shown that STDP is sufficient to achieve some impressive forms of learning [42]–[44], and we therefore use it as a major source of inspiration.

It is insightful to compare ideas coming from neuroscience with the algorithms used in the field of machine learning for processing Big Data. Synaptic plasticity, and in particular STDP, shares features with learning rules used by various machine learning paradigms. For example, STDP depends only on the activity of the presynaptic and of the postsynaptic neuron. This locality feature is similar to the learning rule of Restricted Boltzmann Machines, the first system used to demonstrate deep unsupervised learning [45]. Some learning rules, however, are fundamentally non-local. This is the case for backpropagation, traditionally used in artificial neural networks [46] and in the popular convolutional neural networks

[2], [47], [48]. Other machine learning paradigms such as Support Vector Machines use fundamentally different principles than STDP and neurons.

In this work, we program memory devices to implement a simplified version of STDP. Several proposals exist for implementing traditional STDP with purely CMOS circuits [49]–[51]. Numerous other works propose implementing it directly when programming memory devices [5], [11], [13], [14], [18], [52]–[55]. In contrast with purely CMOS solutions, this brings a higher synaptic density, offering the possibility of massive connectivity between neurons as in the human brain. Even more importantly, the memory devices offer non-volatility to the synapses, preserving their state without time degradation, and thus offer instant ON / OFF operation to the system. In these proposals, the memory devices are hybridized with CMOS neurons, which do not require non-volatility. This idea is made feasible by the tremendous progress on implementing neurons with CMOS [30], [56], and on the very large scale integration of such neurons [31], [32].

Most proposals for implementing STDP with memory devices possess a relatively high complexity. They usually employ multiple programming pulses [52] or complex analog programming waveforms [11]. In this work, we abandon biological realism for a highly simplified version of STDP [36], shown in Figure 2(b). The simplification results from two major ideas. First, we ignore the analog time dependence of STDP, allowing only two possibilities: increasing or decreasing synaptic weight. Second, we decide that the actual synaptic weight change specifically when a POST spike occurs. At this point, if the presynaptic neuron was active recently, the synaptic weight is increased (synaptic potentiation). In any other case, it is decreased (synaptic depression).

This strategy, which is not mandatory and loses part of the richness of STDP, provides two main benefits for implementation. First, we need only use simple square voltage pulses. Second, the STDP learning rule is applied during clearly defined "programming" phases, which are distinct from system "operation". We can thus design a system such that nothing else can happen during the programming phase, making circuit design significantly less complex. We show in section III how this works in practice, and that this extremely simplified version of STDP can allow a system to learn sophisticated inference.

The precise implementation of STDP, and the exact behavior, depends on the chosen device technology. Different emerging memory devices have unique physical and qualitative behaviors, as illustrated in Figure 3. While such disparities do not have many fundamental consequences when the devices are used as binary memories, they become very apparent, however, when the devices are used as plastic synapses. In this context, a complete understanding of their operating principles and electrical characteristics is necessary.

Nevertheless, our approach adapts to many kinds of memory devices, which can have multilevel as well as binary memory capability.

B. Implementation with a Multilevel Memory

1) "Cumulative" Memristive Device: We first consider the case of memristive devices. Many emerging memory technologies have some form of multilevel capability, and this was a core idea of the original memristor proposal [4] illustrated in Figure 3(a). Devices with multilevel behaviors similar to the memristor idea function based on different physical phenomena: ionics [5], [14], [61], ferroelectrocity [20] and magnetic domain wall motion [62]. In these devices, it is possible to increase or decrease the conductance of the device by applying short programming voltage or current pulses. The change is cumulative: if one repeats a programming pulse to a device, its conductance will change further, as is seen in the measurements of Figure 3(a) (reproduced from [5]). This behavior is reminiscent of synaptic plasticity, and it is thus natural to compare memristive devices with synapses.

The way these memristive devices are used is straightforward. The conductance of the device encodes the "strength" of the synapse, or synaptic weight. Applying a brief positive voltage pulse that is higher than a programming threshold (V_{T+}) increases the conductance of the device. Applying a brief negative voltage pulse that is lower than a programming threshold (V_{T-}) decreases the conductance of the device. To obtain a simplified STDP analogue, we need only apply cleverly designed programming voltage pulses, as illustrated in Figure 4(c). When the postsynaptic neuron fires, it applies a waveform to its synapses, while the presynaptic neurons that were active recently simultaneously apply a simple voltage pulse. Synapses presenting only the postsynaptic waveform have their conductance reduced. Synapses that experience both the postsynaptic waveform and the presynaptic pulse have their conductance increased.

This implementation (analogous to Figure 3 in [63]) clearly separates phases in which the device is used for transmitting information, from phases in which it is programmed and learning occurs. It is only possible if, when STDP programming occurs, the system does not need the synapses to transmit information. This is typically the case in the Winner Takes All-type systems studied in section III. It is also possible to avoid this separation between transmission and programming by making input pulses very long. This is illustrated in Figure 2 of [63], and in various other proposals such as [11], [53], [55]. Such a scheme, however, makes the design of the synapses driving circuitry harder [64] and thus leads to a significant energy consumption due to the input pulses.

2) Phase Change Memory: In a phase change memory (Figure 3(b)), a small volume of material can switch between a high resistance amorphous state and a low resistance crystalline state [17], [58], [65]. A characteristic of phase change memory when compared with more conventional memristive devices is its unipolar nature: positive and negative voltages are equivalent. While the transition from the amorphous to the crystalline state exhibits the desirable cumulative behavior found in memristive devices, as is seen in the measurements of Figure 3(b), the reverse transition from the crystalline to the amorphous state is not cumulative [58]. The programming current determines the final state, and if one repeats the same programming pulse twice, the second pulse has no effect. This is a serious issue for implementing a functional synapse with

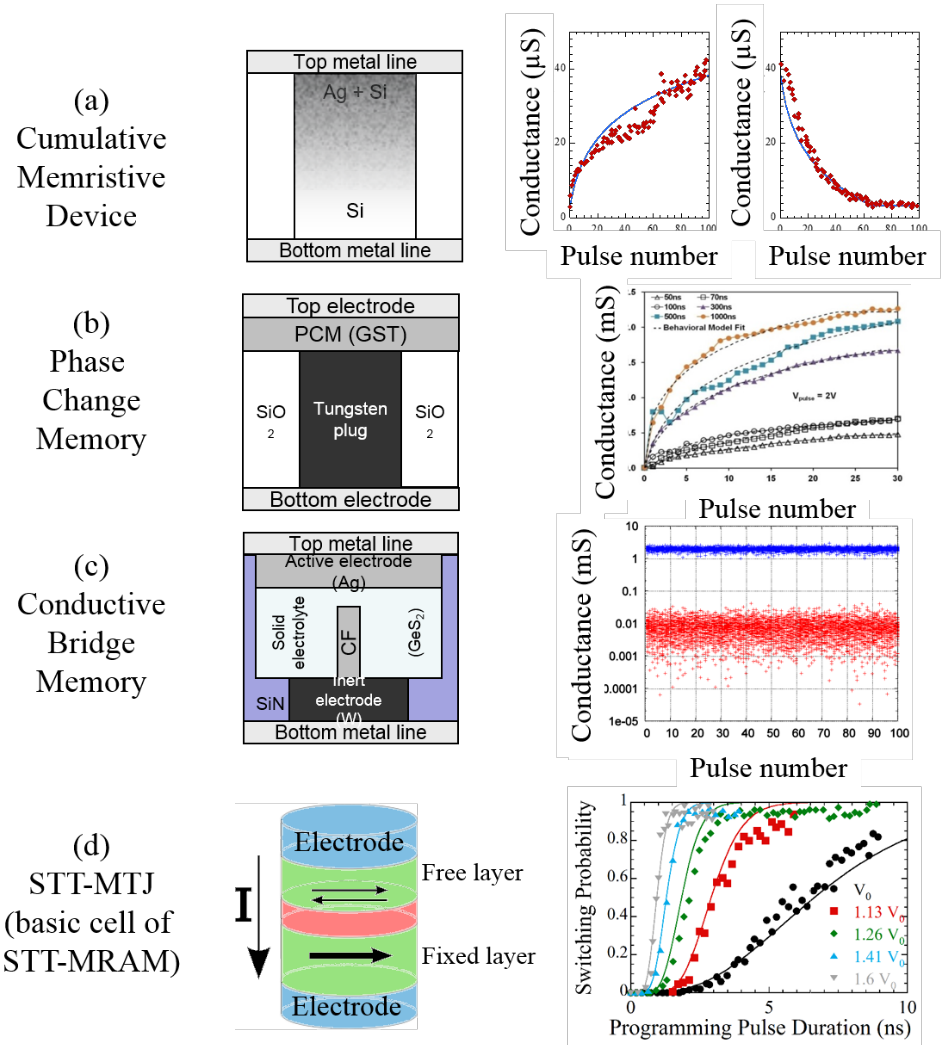

Fig. 3. Different nanodevices and their behavior. (a) Cumulative memristive device (measurements from [5] and model from [57]). (b) Phase Change Memory (measurements and model from [58]). (c) Conductive Bridge Memory (measurements and model from [59]). (d) Spin Transfer Torque Magnetic Tunnel Junction. (measurements and model from [60]).

phase change memory.

A possibility to regain cumulativity in both directions is to associate two PCMs in the "2-PCM" structure described in [65]. Additionally, it should be noted that PCMs suffer from a drift issue that makes low resistive states partially unstable. This does not, however, seem to forbid neuromorphic applications [66].

C. Adaptation to Other Device Physics – Stochastic Synapse

1) Conductive Bridge Memory: In most CMOS-compatible conductive bridge memories (CBRAM) [59], [67], both directions of programming are noncumulative (Figure 3(c)). It is possible to provide multilevel memory by carefully choosing the programming current, but applying the same programming pulse multiple times has no further effect (some academic technologies may exhibit cumulative effects or more complex behaviors).

We have therefore proposed an alternate method for using CBRAMs [59], [67]. We chose to treat the CBRAMs as binary devices, and to replace the progressivity of multilevel operation with a progressivity resulting from probabilistic programming. The idea is simple: when STDP occurs, the memory device only has a *probability* of changing its conductance. When learning occurs, with traditional STDP, all synapses change their conductance by a small amount. Here, however, a fraction of synapses whose weight should slightly increase change their weight entirely, while a fraction of synapses whose weight should slightly decrease change their weight entirely, but most synapses' states do not change. Stochastic STDP is illustrated in Figure 2(c). We compare this form of STDP to traditional STDP in section III and show that it can be quite powerful. In theoretical works, the idea of using stochastic synapses instead of deterministic ones (in a broad sense) has also been proposed with supervised neural networks [23], [68], [69].

An important question is how to actually implement this probabilistic behavior. One obvious solution is to use pseudo-random number generators to achieve stochastic programming. A more ideal option is to have some kind of intrinsic probabilistic effects inherent to the devices. Both options are

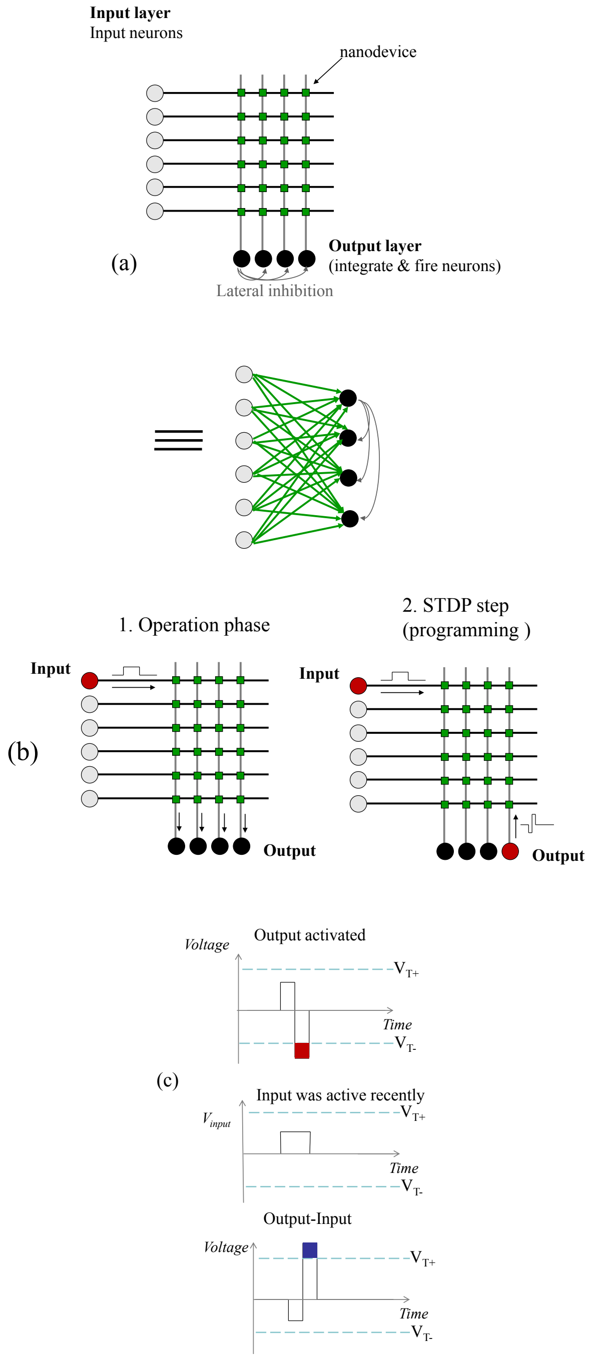

Fig. 4. (a) Simplified crossbar architecture of the inference engine, and its equivalent neural network. (b) Representation of the two phases of the system: operation and STDP. (c) Voltage pulses used to implement STDP with cumulative memristive devices.

possible with conductive bridge memory, but the second option may prove more difficult to control. Additionnally, it has not been proven experimentally that the intrinsic probabilistic effects in conductive bridge memory are sufficiently random [59], [67].

2) Spin Transfer Torque Magnetic Tunnel Junction: One final remarkable case is the spin transfer torque magnetic tunnel junction (STT-MTJ, Figure 3(d)) the basic cell of Spin Transfer Torque Magnetic memory (STT-MRAM) [6], [70], [71]. STT-MTJs may be the ideal technology for the stochastic STDP approach. Unlike the previously mentioned technologies, this device is truly binary: it has only two memory states, the high resistive anti-parallel (AP) state and the low resistive parallel (P) state. A striking feature is that switching is fundamentally stochastic, as observed in the measurements of Figure 3(d). This effect is also seen in CBRAMs and other kinds of memory when using weak programming pulses [28], [59], [72], [73]. It is, however, better understood and controlled in STT-MTJs than in CBRAMs, as it emerges from the basic MTJ physics [6], [8]. Furthermore, STT-MTJs switching can be modeled with comprehensive analytical equations [60], which we use in the present work.

While this stochastic switching is deleterious for applications such as memory cells, and necessitates the use of long programming pulses to ensure reliable switching [74], it can fortuitously be used to generate true random numbers, as their quality has already been verified with regards to true random number generator standards [75].

III. Example of Inferences Made with STDP-Type Programming

A. Problem of Classification

1) Basic Principle: We now describe how STDP-type programming can allow a system to classify images, i.e. to infer in which class an image belongs. In fact, the system not only performs the inference, but also identifies and creates the different categories by which the images can be categorized. These categories constitute the values of a “latent variable” of the data in the inference language.

The canonical dataset for machine learning is the MNIST handwritten numbers dataset [46], and its is thus a natural first test for our learning approaches. It consists of 60,000 handwritten digits dedicated to training a system, and 10,000 digits for testing it.

To realize our system, we lay out the memory devices in a conventional memory crossbar structure as illustrated in Figure 4(a), and use CMOS circuits for the input and output neurons. Each input neuron is connected to each output neuron by a memory device. This structure is feasible for devices with strong nonlinearity that limit sneak paths [76]. For other devices, access devices (diode or transistor depending on the technology) might be necessary (as in phase change memory [77] or conductive bridge memory [59]), but the basic principle remains the same.

It is straightforward to understand how such a system may learn. We present the handwritten digits converted to spikes as the input to the system, each input neuron corresponding to one pixel of the image. Alternative possibilities to encode the pixel values as spikes are described in [63], and do not significantly affect system-level performance. These input spikes give rise to currents as they are applied to the memory devices, the magnitude of these currents being directly determined by the conductances of the devices. These currents I_{out} are received by the output neurons, which act as leaky Integrate-and-Fire neurons [30]. That is, they have an internal state variable X that evolves according to a first order differential equation:

$$\tau \frac{dX}{dt} + gX = I_{out}, \tag{1}$$

where τ is an integration time constant associated with the neuron, and g a real constant the value of which approaches one.

When the variable X of one of the output neuron reaches a given threshold, this output neuron spikes and X is reset to zero during a "refractory period". Many efficient options (analog or digital) exist to implement leaky Integrate-and-Fire neurons with CMOS [30], [56]. When the output neuron spikes, it also inhibits the other output neurons during an inhibition period. This can be efficiently implemented by a nearest neighbor scheme like the diffuser network used in [49].

After an output neuron has spiked, the STDP step also occurs (Figure 4(b) and (c)). It affects all the synapses connected to the output neuron that spiked. In STDP, the synapses that were active recently become stronger; that is, the synapses change such that the output neurons will spike faster in response to a similar input. All other synapses connected to the output neuron become weaker; that is, if an input substantially differs from the one which caused the output neuron to spike, the neuron will be less likely to activate, and thus leave more time for the other neurons to activate. The output neurons thus become specialized to a particular type of input. We show in the next sections that this approach works impressively well in practice, for different types of problems.

Interestingly, this process is entirely unsupervised: every output neuron progressively becomes specialized to a class of patterns that continuously return. These categories that implicitly separate the input presented constitute a latent variable, using the language of inference. Through the process of learning, our system is able to identify the values of latent variable behind the inputs, and to classify the input among them.

2) Performance on the MNIST Dataset: This process is observed when we present the MNIST dataset to the system. This example is taken from [16], [63], and is based on system level simulation [78]. This kind of simulations, based on a specialized simulator written in the C++ programming language, includes comprehensive physical models of the memory devices, but simulates their peripheral circuits functionally. This allows simulations to be considerably faster than SPICE simulations, and to simulate realistic full scale applications. The detailed physical models of the memory devices allow us to understand how device properties translate in terms of learning the applications. In particular, it is possible to include all imperfections seen on real devices and to evaluate the resilence of the applications. On the other hand, some effects due to peripheral circuits (like neuron variability) need to be simulated in a highly abstracted way.

As memory devices, we are inspired from a cumulative memristive device [5]. The device model, fitted from experiments, is presented in [57] and based on realistic parameter values following [5]. This model includes precisely the dependence of conductance change to the state of the memristive devices. It allows including effects of device mismatch, as explored in section V. Discussion of the required precision of the devices, and of how controlled the multilevel behavior of the device needs to be appears in [63].

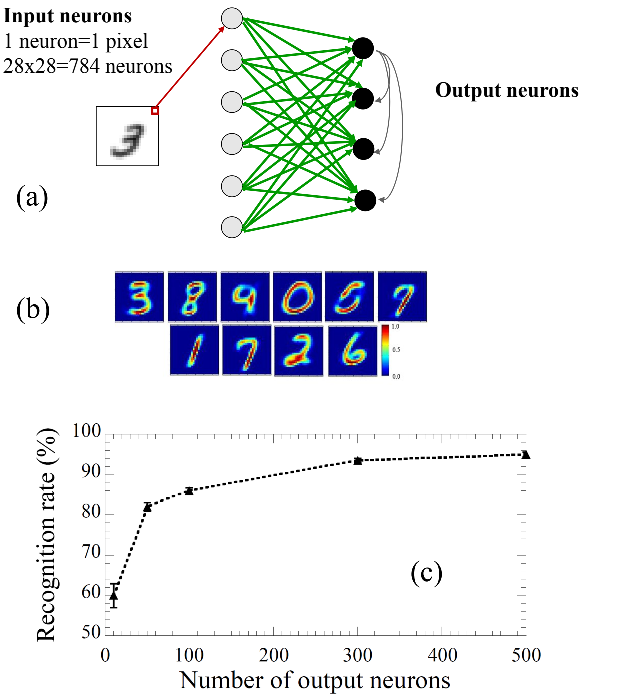

Fig. 5. Results on MNIST recognition with cumulative memristive devices. (a) System topology for processing MNIST data. (b) Representation of the final conductance of memristive devices, at the end of learning, with ten output neurons. Each image is a 2-D representation of the devices connected to one output neuron. (c) MNIST recognition rate as a function of number of output neurons. Error bar is one standard deviation.

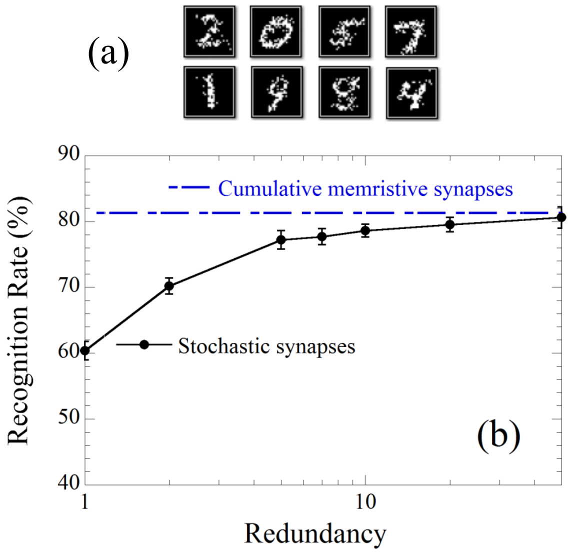

Fig. 6. (a) Representation of the final conductance of stochastic synapses, at the end of MNIST learning. Each image is a 2-D representation of the devices connected to one output neuron.Black: high resistance state, white: low resistance state. (b) Recognition rate as a function of device redundancy (number of devices connecting each input to each output), in a situation with 50 output neurons. Horizontal line: recognition rate with cumulative memristive devices in the same situation.

We first consider a simple system with ten output neurons, according to the basic architecture of Figure 5(a). The final post-learning weights of each synapse connected to an output neuron is plotted as a two-dimensional image in Figure 5(b). We can see that each output neuron has specialized in a type of digit, and has learned its distinctive features: the loop of the digit two, the bars of the digit eight, etc. This suggests that the system has played its role as an inference engine, it has recognized the different digits, which constitute the latent variable behind images of handwritten digits.

If we identify the correspondence between digits and output neurons, we can translate the performance of the system on the test dataset as a recognition rate. It reaches 60%, which is not an impressive recognition rate by machine learning standards, although it is significantly better than random choice (10%). This results from the fact that there are numerous ways to hand-write the same digit. The true latent variable behind the MNIST dataset thus represents the various typical handwrittings of all digits. As can be seen in Figure 5(b), the system learned two handwritings of the digit seven, yet no handwriting of the digit four. Thus, we need more than ten output neurons for successful digit recognition.

We therefore performed simulations with various numbers of output neurons. At the end of learning, using a limited set of the dataset, we automatically identify the digit to which each output neuron corresponds. This step may be performed using simple counting or by another spiking neural network as described in [79]. We then tested the system on the MNIST test dataset. This allows us to plot recognition rate as function of the number of output neurons, as shown in Figure 5(c). For 300 output neurons, the recognition rate reaches 93.5%. This is far from the best results obtained for the dataset, which used millions of adjustable parameters and an augmented dataset, and where the recognition rate can reach 99.7% [80]. However, our unsupervised results compare well to supervised neural network with back-propagation and the same number of adjustable parameters, which have a recognition rate of 95% [46]. The supervised approach requires knowing which digit corresponds to every image of the training dataset, while the unsupervised approach requires knowledge of only a small subset. This is an important feature in the era of Big Data, where we have access to massive amounts of unlabeled data.

A big advantage of the MNIST benchmark is that it is provides a natural metric to compare potential devices as artificial synapses. In particular, it is of special interest to compare cumulative memristive synapses with stochastic synapses such as stochastically programmed conductive bridge memory or the intrinsically stochastic STT-MTJs. In Figure 6, we show our results for a stochastically programmed conductive bridge memory instead of the cumulative memristive devices. The device model, described in [67], is fitted on experiments. In particular, it includes the experimental intrinsic cycle-to-cycle dispersion on device conductance: each time a device is programmed to the ON of OFF state, its conductance is varied.

Figure 6(a) shows the final states of the memory devices for eight of the output neurons. This appears similar as Figure 5(b), but with binary (black and white) values instead of a color spectrum representing multilevel weights. The system has 50 output neurons, and would have a recognition rate of 82% with cumulative memristors. If we replace every cumulative memristive device by just one stochastic device, the recognition rate is a little better than 60%. However, it is possible to obtain equivalent recognition by introducing redundancy. Figure 6(b) shows the recognition rate as a function of number of stochastic devices replacing one cumulative memristive device. We see that if we replace a cumulative memristive device by 5 stochastic devices, a recognition of 77.2% is achieved, and with 7 we reach the 78.0%. As we show in the next subsection, the required level of redundancy is extremely problem-dependent and can be much smaller for other tasks.

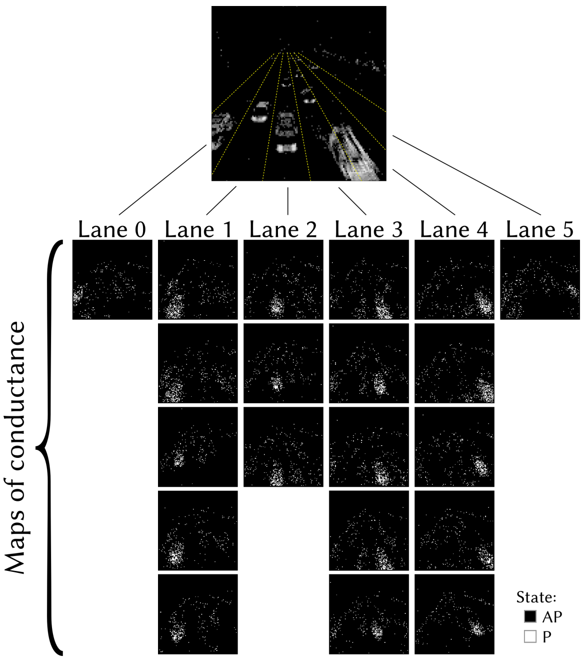

Fig. 7. Simulation of the car detection task using STT-MTJs. Top: part of the stimulus, obtained by averaging the input spike-based video over 30 ms. Bottom: final weights of the stochastic STT-MTJs. Each image corresponds to one output neuron. The outputs neurons are classified by the lane to which they specialize.

B. Detection within Dynamic Data

Beyond image classification, the time-dependent nature of STDP makes it particularly appropriate to learn inference on dynamic data. First, we take an example of video processing [77]. We used a video acquired from a bioinspired dynamic vision sensor [81], which naturally produces spikes, analogous to our retina. The video shows vehicles moving in front of a camera on a freeway in Pasadena, CA, USA, and is freely available online [82]. A 2-D presentation of the spikes, integrated over 30 ms, is shown in the top picture of Figure 7. Here, the latent variable behind the video represents the vehicles passing on the six lanes. If our system learns to recognize them, it naturally becomes a proper vehicle counter.

To learn this task, we use the same system as for the MNIST case. Each input neuron is connected to one pixel of the camera. Only the actual values for the neurons' threshold, refractory and inhibitory periods need to be changed for this new problem. The results are impressive: in the case of cumulative memristive devices [65] or of 2-PCM structure [77], the system becomes a vehicle counter with detection rate higher than 95% on all lines except the two outer lane where very few cars are passing. Less than ten false positives occur during the 80 seconds video. Cumulative memristive devices were modeled with the same model used as for MNIST recognition [57], and 2-PCM structures with a variation of this model fitted on experiments [58]. Including the drift effects seen in the high resistance states of PCM did not affect the result. As a comparison, the best result on the same dataset, using a neural network with double precision analog weights obtains a detection rate of 98.1% and 9 false positives [77].

The results with stochastic devices are even more interesting. Figure 7 shows the results with STT-MTJs, where we replaced every cumulative memristive device by only one binary STT-MTJ. STT-MTJs are modeled with a comprehensive physics-based analytical model which reproduces full magnetic simulations [60]. This models allows including various device imperfections (device mismatch, dependence to temperature) in a physical way. In terms of detection rate and number of false positives, in simulations employing the STT-MTJ analytical model, we still obtain recognition rates higher than 95% on the four inward lanes, and less than ten false positives [83]. Unlike the MNIST case, redundancy was not necessary to reproduce the performance obtained with cumulative memristive device with stochastic binary devices.

As the car detection constitutes a practical task, it is insightful to estimate the energy that needs to be used by the system to program the memory devices. These estimates solely concern the programming energy at the device level (peripheral circuitry is ignored). With a conservative Phase Change Memory technology (used in a 2-PCM structure), the energy consumption during learning has been estimated to 110 μW. This power consumption value scales with the technology node and could be as low as 100 nW on advanced PCM [65]. With stochastic conductive bridge memory, energy consumption has been estimated to 74 μW [59]. With STT-MTJs representative of a 45 nm technology, energy consumption has been estimated to 4 μW using long programming pulses with very low voltages [71], [83], and can be as low as 180 nW using shorter programming pulses with higher voltages (0.46 V) [71]. These values support the potential of our bioinspired programming approach for low energy computation. They are much smaller than the power consumption of processors capable of machine learning. However, designing the full system is necessary to evaluate the final global power consumption. Once the car detection task has been learned, we may choose to either continue STDP so that the system can adapt to changes in the inputs, or to deactivate STDP to save programming power.

The system can also be used to process auditory data. In this case, we implement a collection of filters inspired by the cochlea [84], and the result of each filter is connected to one input neuron. We then present inputs consisting of audio noise, in which repeated patterns have been inserted. After learning, every output neuron becomes sensitive to one of the repeated patterns. The final sensitivity is similar to that demonstrated by humans given the same problem. The details of this task are given in [59].

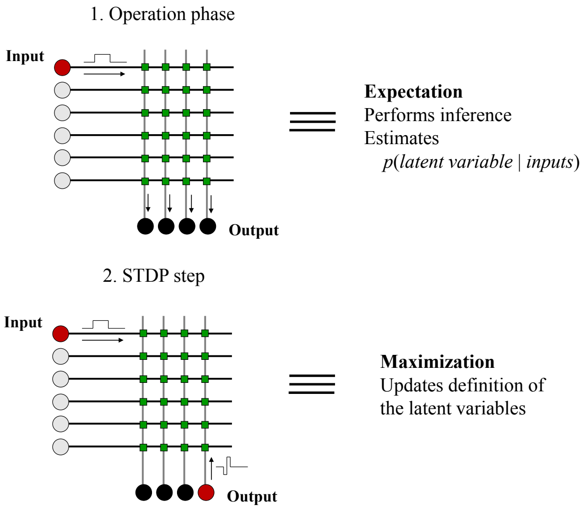

Fig. 8. Equivalence between the inference engine and the Expectation-Maximization algorithm.

This task can also be implemented either with cumulative memristive devices or stochastically programmed ones. However, to achieve similar sensitivity with both implementations, each cumulative memristive devices needs to be replaced by three stochastically programmed devices.

These three examples illustrate that in many situations, stochastically programmed binary devices can function equivalently to cumulative memristive devices. However, the quantity of redundancy to introduce in the binary case varies greatly. No redundancy is needed for car counting, one cumulative device needs to be replaced by three stochastic devices for auditory pattern detection, and by five for roughly equivalent MNIST character recognition.

IV. Theoretical Analysis of the Inference Engine

A. Link with Bayesian Inference

We have seen that simplified STDP-type programming of memory devices can allow the system to learn sophisticated inference, in very different situations. In the field of computational neuroscience, a very powerful theory exists that can explain this. In several works, Maass has studied how systems of spiking neurons can be used to perform Bayesian computation, in the presence of high noise and stochasticity [85]. The most relevant work for our discussion investigates how a specific form of STDP can lead to a form of optimal Bayesian inference [36], by approximating the powerful machine learning algorithm of Expectation-Maximization.

The authors of ref. [36] use a simplified version of STDP, which has the same graph as the STDP that we proposed for memory devices (Figure 2(b)). However, the exact impact of STDP steps on the weight of the synapses differs from what we have considered previously. We can introduce w the synaptic weight, δw_+ the weight increase when a presynaptic spike preceded a postsynaptic spike, and δw_- the weight decrease in other situations. Nessler *et al.* considered:

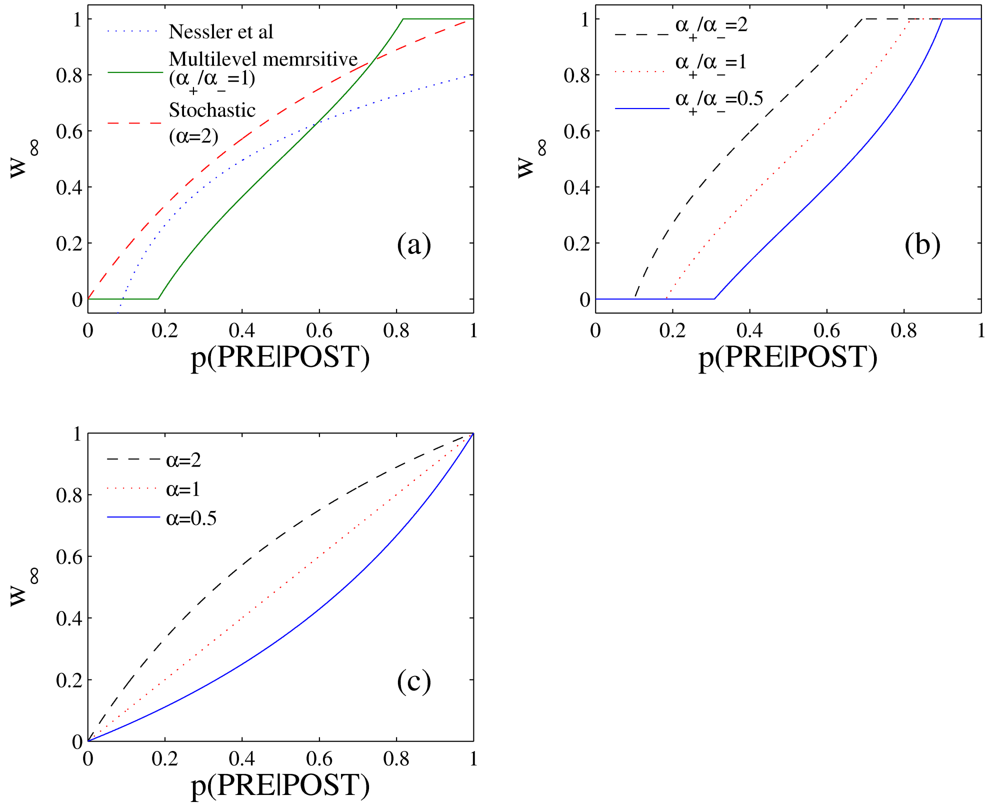

Fig. 9. (a) Final weight as a function of $p(PRE|POST)$ in Nessler theory (equation 4), compared with cumulative memristive devices (equation 7) and final probability of a stochastic synapse being in the state 1 (equation 8). (b) Final weight as a function of $p(PRE|POST)$ in with cumulative memristive devices for different α_+ and α_- values. (c) Final probability of a stochastic synapse being in the state 1 as a function of $p(PRE|POST)$ for different α values.

$$\begin{aligned}\delta w_+ &= C\exp(-w) - 1 \\ \delta w_- &= -1, \end{aligned} \tag{2}$$

where C is a real constant greater than one.

The exact comparison with the behavior of physical memory devices will be discussed in the next subsection. Based on equation 2, it is is straightforward to evaluate analytically which values the synaptic weights approach at the end of a learning process. We introduce

$$\begin{aligned} & p(PRE|POST) = \\ & \qquad p(t_{PRE} \in [t_{POST} - t_{STDP}; t_{POST}] | t_{POST}), \end{aligned} \tag{3}$$

the probability that when an output neuron spiked, a synapse spiked in the STDP window t_{STDP} preceding the spike. Then we can show that the final weight w_∞ that this particular synapse will approach during the learning process is

$$w_\infty = \log p(PRE|POST) + \log C. \tag{4}$$

This simple result has deep implications. Nessler *et al.* have shown that this allows the system to perform an approximation of Expectation Maximization, an extremely powerful machine learning algorithm [86]. The process is illustrated in Figure 8. The two phases in our system directly correspond to the ones of the Expectation Maximization algorithm. The operating phase where synapses transmit information from input to output until an output neuron spikes correspond to Expectation steps, the execution of inference. The STDP programming steps correspond to Maximization steps, the optimization of the latent variable.

This theoretical study provides insight into how our inference engine learns and performs inference. However, the physical synapses that we have studied do not correspond to equation 2. We now consider the details regarding the importance of this difference.

B. Link between Device Physics and Learning

1) Cumulative Memristive Synapses: We first consider the case of cumulative memristive devices. We use a simple model of the conductance increase and decrease [57], which fits the measurements of the classical cumulative memristors of [5], and can also be used to model the 2-PCM structure [58], [65]. For the sake of simplicity, we use normalized units $w = G/G_{MAX}$, and assume that minimum and maximum conductances are 0 and 1. We identify normalized conductance with a synaptic weight w. The device model of [57] then simplifies to

$$\begin{aligned} \delta w_+ &= \alpha_+ \exp(-\beta_+ w) \\ \delta w_- &= \alpha_- \exp(-\beta_-(1-w)). \end{aligned} \tag{5}$$

α_+ and α_- represent by how much the conductance of the memristive device changes when a programming pulse is applied. Smaller α values lead to more analog behavior. β_+ and β_- model the dependency of this conductance change with the state of the memristive device. β values of the order of 3.0 can model the devices of [5], [58], [65].

Additionally, the weight is bounded by a minimum (0) and maximum (1) conductance.

Under these conditions, we can show that the final weight of this particular synapse approaches

$$\begin{aligned} w_\infty =& \frac{\beta_-}{\beta_+ + \beta_-} \\ &+ \frac{1}{\beta_+ + \beta_-} \log \frac{p(PRE|POST)}{1 - p(PRE|POST)} \\ &+ \frac{1}{\beta_+ + \beta_-} \log \frac{\alpha_+}{\alpha_-}, \end{aligned} \tag{6}$$

with w_∞ being additionally bounded between 0 and 1. The derivation appears in Appendix A.

In the case where β_+ and β_- are equal, this simplifies to

$$w_\infty = \frac{1}{2} + \frac{1}{2\beta} \log \frac{p(PRE|POST)}{1 - p(PRE|POST)} + \frac{1}{2\beta} \log \frac{\alpha_+}{\alpha_-}. \tag{7}$$

This equation is reminiscent of equation 4, but a significant difference is that w_∞ appears to approach infinity when $p(PRE|POST)$ approaches one. However, since the weight of a physical device is bounded between 0 and 1, this divergence does not actually occur. When considering and putting practical values into equation 6, it becomes in fact remarkably similar to equation 4. This is shown clearly in Figure 9(b), where equation 6 is plotted for different values of α_+/α_-, and the value for β is taken from real devices [5]. This suggests that our inference engine with cumulative memristive devices may work by an approximation of Expectation-Maximization.

Interestingly, the curves corresponding to differing values of α_+/α_- (2.0, 1.0 and 0.5) are qualitatively similar. This is in agreement with the fact that when simulating the problems of section III, the value of α_+/α_- is not a sensitive parameter. For example, on the car counting task, and with cumulative memristive devices, the best recognition rate on the four inward lanes (99%) is obtained with $\alpha_+/\alpha_- = 2.0$. With $\alpha_+/\alpha_- = 1.0$, the recognition rate on the four inward lanes is only slightly reduced (97%). This result has important implications when dealing with nanodevices. The parameters associated with learning do not need to be too fine-tuned for the system to be able to learn tasks.

Additionally, we notice that only the ratio α_+/α_- appears in equation 6, not the actual α value. This is also consistent with the research in section III. This does not mean, however, that the actual α values are entirely insignificant, as they directly affect the speed of learning.

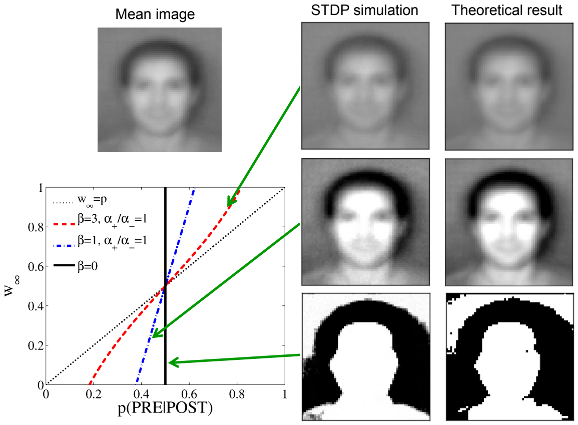

Fig. 10. Top left picture: mean image of the faces presented. Right pictures: representation of the final weights in a system with one neuron, to which photographs of faces was presented, as obtained in simulation, and as expected from theory. With cumulative memristive devices and top: $\beta = 3$, $\alpha_+/\alpha_- = 1$, middle: $\beta = 1$, $\alpha_+/\alpha_- = 1$, bottom: $\beta = 0$. Graph: Final weight as a function of $p(PRE|POST)$ in these three situations.

Finally, we should note that the β value has considerable impact on the shape of the w_∞ curves. To illustrate this, we simulated a network with only one output neuron, to which we presented static photographs of faces, and repeated the simulation for devices with different β values. The resulting synaptic weights, organized as a 2-D picture, as well as the corresponding w_∞ curves appear in Figure 10. We can see that with a β value of 3.0 (which is close to what is observed in the devices of [5], or the 2-PCM structure [65]), the final weights are very analog, and approach the mean of all the presented faces. By contrast a β value of 1.0 produces a more binary map, amplifying what is distinctive about a face. A β value of 0 leads to an entirely binary map separating pixels where $p(PRE|POST)$ is lower and greater than 0.5. This corresponds well to what would be expected from w_∞ as a function of $p(PRE|POST)$ curves. This means that, depending on device, very different kinds of learning can thus be envisioned.

In summary, we have observed a remarkable insensitivity to relative steps of potentiation and depression, as well as to the actual value of these steps (α values). We have observed that the devices with different dependences of steps with actual values of the conductance (β values) can have different learning characteristics.

2) Stochastic Synapses: We now consider the case of stochastic programming, which we introduced in particular for conductive bridge memory and STT-MTJs.

We introduce p_+ the probability for a synapse to switch from low conductance (“0”) to high conductance (“1”) when a presynaptic spike occured before the postsynaptic spike, p_- its probability to switch from high conductance to low conductance in the other situations, and $\alpha = p_+/p_-$. At the end of the learning process, we show in Appendix B that the probability of a synapse to be in the high conductance state is

$$w_\infty = \frac{\alpha p(PRE|POST)}{1 + p(PRE|POST)(\alpha - 1)}. \quad (8)$$

If p_+ and p_- are equal ($\alpha = 1$), w_∞ reduces to $p(PRE|POST)$. As can be seen in Figure 9(c), the shape of the w_∞ as a function of $p(PRE|POST)$ appears relatively different from those of equation 4, but retains some of its distinctive features. It can be thus expected that learning with stochastic synapses also performs an approximation of Expectation-Maximization in an extremely stochastic form.

When redundancy between stochastic synapses is introduced, w_∞ not only represents the probability of an individual device to be 1, but also a mean value of the weight of the equivalent synapse formed by the ensemble of the stochastic synapses. It is thus natural that the system approximates Expectation-Maximization better, as was seen with the MNIST classification task (Figure 6(b)).

Finally, it is insightful to compare the w_∞ curves for α values ranging from 0.5 to 2. Once again, the curves are qualitatively relatively similar. This result is consistent with our practical observation that the choice of α is not extremely sensitive when solving the actual tasks of section III, although more sensitive than α_+/α_- in the case of cumulative memristive devices. For example, when solving the vehicle counting task with an α value of 1.0 ($p_+ = p_- = 0.1$), the detection rate is 97.3%. With an α value of 2.0 ($p_+ = 2p_- = 0.1$), the detection rate is reduced significantly, but remains high (83.0%). Once again, this is an essential feature for being able to use a system with real devices, where mean switching probability might not be tuned with an arbitrary precision.

V. Robustness of the Inference Engine to Noise and to Device Imperfections

As mentioned in the introduction, device non-idealities may be a fundamental obstacle to the approach that we are proposing. In particular, new memory devices – like almost all nanodevices – suffer from a high level of device variation. Monte Carlo simulations allow us to evaluate and understand the robustness of our inference engine with respect to device non-idealities.

A. Robustness to Device Variability

To study the impact of device variations on the inference engine, we perform Monte Carlo simulations where every device in the system is different, as is routinely done in modern CMOS circuit design. We first consider the case of cumulative memristive devices, and take the example of MNIST recognition. We use a system with only 50 outputs, which without variability has a recognition rate (as defined in section III) of 82%. Recognition rates when variability is introduced are plotted in Figure 11. We introduced variations on the initial states of the devices, on their minimum and maximum conductance, and on the α parameters of equation 5. These α parameters model by how much the conductance of a memristive device changes when it is programmed. The variability is expressed in relative standard dispersion σ/μ. We can see that variations of 10% on all parameters have no impact on the recognition rate, and variations of 25% reduce the recognition rate only by a few percent. Extreme variations of 100% reduce the recognition rate to 70%, but is still higher than random recognition (10%), and it is incredible that the system remains functional with so much variability of the device properties. It should be noted that, in this situation, approximately one third of the devices have an α value of zero in at least one direction, and are thus practically incapable of learning. Additionally, devices where the minimum and maximum conductance appear inverted due to device variations are also considered as devices incapable of learning.

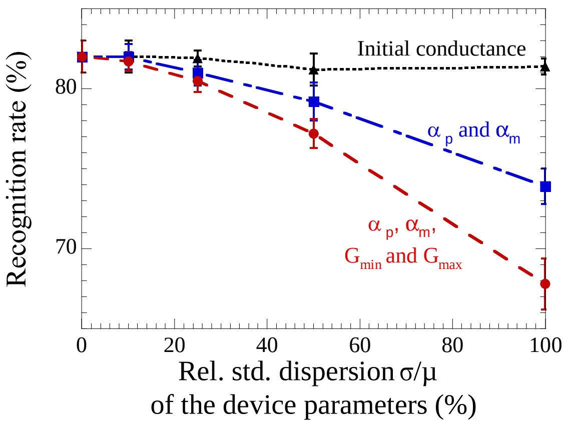

Fig. 11. Impact of device variations, on the problem of MNIST recognition with cumulative memristive devices, with 50 output neurons. Recognition rate as a function of relative standard deviations on device parameters. Triangles: only the initial conductance. Squares: only the α_+ and α_- parameters of equation 5. Circles: initial conductance, α_+ and α_-, minimum and maximum conductance. Each simulation was repeated ten times. The error bar indicates one standard deviation.

A similar robustness was observed in the task of car counting. With variations of 25%, the detection rate is identical to that seen in the case of no variations. We call such robustness to device variability with respect to traditional uses of memory near-immunity. More in-depth discussions of these results appear in [63].

We also performed Monte Carlo simulations for the case of stochastic devices. In this implementation, device variability affects minimum and maximum conductance. If the stochastic effects directly emerge from the device physics, like we proposed in the case of STT-MTJs, the probability to switch p_+ and p_- (as introduced in section IV) will also be variable, possibly dramatically. This is illustrated in Figure 12(a), which plots histograms of the low (P) and high (AP) resistance states values, and the corresponding switching probability, when synaptic variability is introduced. It is observed that due to STT-MTJ device physics, switching probability are more disperse than the states' resistance. The synaptic variability introduced corresponds to relative standard deviation (one-sigma) on resistance of the P state and tunnel magnetoresistance TMR ($TMR = (R_{AP} - R_P)/R_P.$). The fact that resistance of the P state and TMR are varied independently is inspired by experiments. A typical value for the synaptic variability in experimental demonstrators is 5% [87], [88].

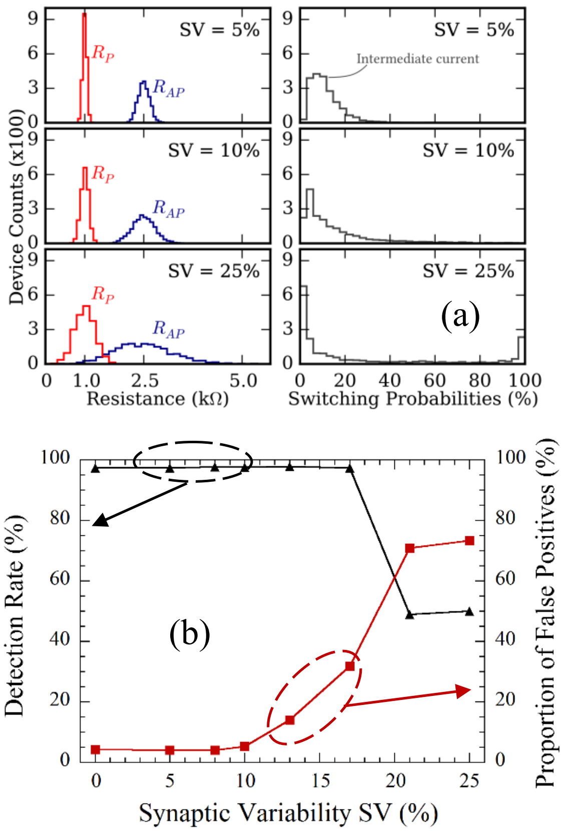

Fig. 12. (a) Histograms representing the values of P and AP states resistance of 2,000 STT-MTJs (left subfigures) and switching probabilities (right subfigures) when synaptic variability (SV) is introduced. Switching probability is 10 % in all cases for SV = 0. From top to bottom, synaptic variability SV is 5 %, 10 % and 25 % of relative standard deviation (one-sigma) on resistance of the P state and TMR. (b) Detection rate and proportion of false positives as a function of synaptic variability for the task of car counting with STT-MTJs.

Once again we observed spectacular robustness to device variations. For example, let us consider the case of car counting with STT-MTJs. As observed in Figure 12(b), with device variability of 10% of the STT-MTJs (higher than what is seen is experiments), which directly translate to variability of 61% on the p_+ and p_- parameters, the same detection rate and number of false positives is observed as with no variability [71].

B. Roots of the Robustness to Device Variability

The extreme robustness of our inference engine to device variation is impressive and an understanding of its fundamental origin is instructive for nanoelectronic design. We see two basic elements for this robustness: the unsupervised nature of learning, and the diversity of synapses that can approximate the Expectation-Maximization algorithm.

First, the fact that the system learns in an unsupervised way is an important asset to tolerate variations. When initialized, the neurons are not specialized, and respond more readily to the patterns they are naturally capable of learning. We can for example consider a specific input pattern. If some synapses associated with input neurons fundamental to this pattern do not work, then the output neurons of these synapses will likely learn another input pattern. In that sense, a reasonable device variability is not deeply troublesome for the system. It may even be considered as a feature that precipitates the beginning of the learning process.

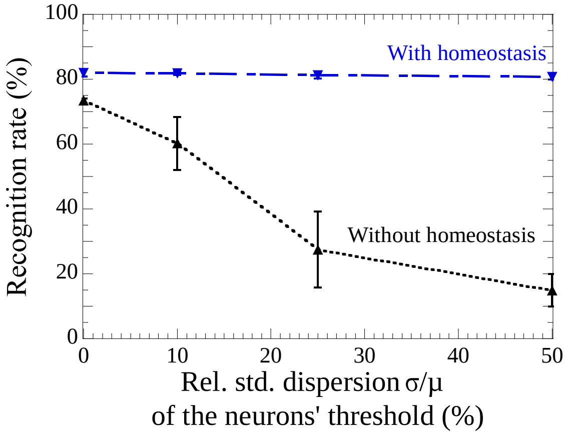

Fig. 13. Impact of homeostasis using the MNIST recognition task, with cumulative memristive devices, where 50 output neurons were used. Recognition rate as a function of variability on neurons' threshold (expressed in relative standard deviation) with and without homeostatic mechanisms. The results are described in detail in [63].

A second component of robustness to variability can be gathered from the theoretical analysis of section IV. For the case of cumulative devices, we have seen that the curve of w_∞ as a function of $p(PRE|POST)$ depends only on the ratio of α_+ and α_-, and that its shape does not qualitatively depend dramatically on this ratio. Similarly, in the stochastic synapses case, the curve of w_∞ depends only on the ratio of p_+ and p_- and its shape does not exhibit significant qualitative dependence on this ratio. This suggests that variable synapses will still manage to perform their task even if they learn through completely different manners. This also suggests that the analysis of section IV can be an effective way to assess if a particular technology will give rise to a robust inference engine.

C. Robustness to Other Issues

First, we should mention the one effect to which our inference engine is not robust: the performance of the system is strongly affected by variability between the neurons [63]. The neurons with lower threshold are activated frequently and prevent the neurons with higher thresholds from learning patterns. Neuron variability is an issue if the neurons are implemented with analog circuits [30]. We proposed a solution in [63], which is to implement a form of “homeostasis” in the neurons (Figure 13). Homeostasis is a bioinspired paradigm that ensures that over long periods, each output neuron spikes a similar number of times. Circuit implementations of homeostasis can cause significant overhead. In digital neuromorphic circuits, homeostasis implementation would be straightforward. It is more challenging for analog neuromorphic systems, since homeostasis requires particularly long time constants. Both pure analog and mixed analog/digital solutions have been proposed [89]. In terms of inference, homeostasis

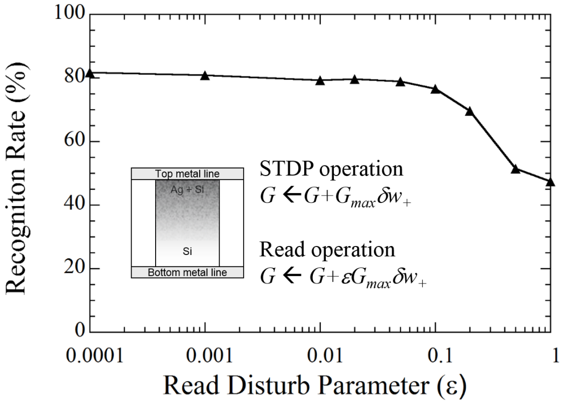

Fig. 14. Impact of read disturb on the task of MNIST recognition with cumulative memristive devices, where 50 output neurons were used. Recognition rate as a function of read disturb parameter ϵ, as defined in the Figure. $\epsilon = 0$ signifies no read disturb.

implements a prior belief that all latent variable values have a significant chance of occurring. It is particularly interesting to note that synaptic variability (which is the primary concern, as it emerges from nanodevices' variability) is naturally tolerated, while neuronal variability requires a correction mechanism. It also stresses the necessity to consider all aspects of a system when studying its robustness.

We can also investigate the robustness to noise in the inputs. It is impressive. For example in the car counting task, for both cumulative memristive devices and stochastic synapses, we can add up to 10% of purely random spikes in the inputs without affecting detection rate and number of false positives. This is caused by the neuronal behavior of the outputs, which naturally acts as a filter for input noise.

Another significant effect in a real system are the transient and potentially local variations of the temperature. Such variations affect all memory devices, but STT-MTJs are among the devices most vulnerable to this effect: since their memory switching is thermally activated, their switching probability depends heavily on the temperature. Therefore, we performed simulations, for the task of vehicle detection, with exacerbated temperature effects: each STDP step, the temperature of all STT-MTJs is chosen randomly. We observed that with temperature chosen randomly with a Gaussian law of standard deviation of $30K$, the detection rate and number of false positives is not affected. Temperature fluctuations act as a form of transient device variations, and the roots of the system's robustness are the same as for static device variations.

Another potential issue is the read disturb effect (the fact that read operation may affect the state of the memory devices). In our scheme, devices frequently transmit spikes. Therefore, even if a read operation has an extremely small effect on the memory state of the device, this effect may accumulate and become significant. Simulations in the case of cumulative memristive devices used for MNIST recognition are presented in [63] and reproduced in Figure 14. The robustness to the read disturb effect is astonishing: if we assume that read disturb affects the device conductance by 1% of the effect of STDP operations, the recognition rate is barely affected. When read operation have 10% of the effect of STDP operations, the recognition rate moves from 82% to a very reasonable 78%. This suggests that the system naturally corrects the drift associated with read disturb. This comes from a very specific feature of our inference engine, which never stops learning. When the system has specialized and is stable, STDP is not deactivated. This allows the system to autocorrect for effects like read disturb, or for a long term evolution of the inputs.

VI. Memory Implications

In this section, we summarize the core ideas that underlie the present work, discuss research approaches that share some features with our proposal, and highlight important open questions.

The present research is based on several fundamental ideas. First, we try to embed memory at the very core of computing, analogous to the brain structure, in order to avoid the von Neumann bottleneck. This vision, which has been previously investigated in several contexts, finds specific appeal today. The emergence of compact, CMOS-compatible, nonvolatile memory offers an ideal technological basis for this concept. Simultaneously, emerging cognitive applications require frequent memory access and suffer from the von Neumann bottleneck. Novel general-purpose or application-specific processor ideas merging computing and memory are therefore being proposed. The level of integration between computing and memory can be comprehensive, in accordance to the original "logic-in-memory" concept [90] approach, or to novel ideas [91]–[93], while some designs keep a stricter separation of computing and memory at a local level [94]. Fusing computing and memory is also at the core of many neuromorphic structures, at the present day usually employing SRAM as memory [31], [32], [49], [50].

In addition to integrating memory and computing, the research reported here incorporates a second idea: to program memory following a bioinspired approach similar to synaptic learning rules. This idea can already be implemented with SRAM [49], [50], but has significant advantages when the memory is based on nanodevices. Programming memory nanodevices in a conventional digital and deterministic way can be wasteful in terms of energy. Bioinspired programming using learning rules, by contrast, can map more closely to device physics, and exploit the complex intrinsic behaviors seen such as multilevel memory and stochasticity. A drawback is that nanodevices used in this fashion exhibit nonidealities like device variations. We have tried to show in this work that bioinspired architectures can feature an intrinsic resilience to these nonidealities, and are therefore able to benefit from the intrinsic features of memory devices.

Other research explores this idea of bioinspired programming. Most of this research attempts to precisely map bioinspired programming to biological models, with the aim of providing a direct correspondence between neuroscience and nanoelectronics research. Here, we suggest that highly abstracted bioinspired programming may already achieve useful features. The balance between biological abstraction and biomimetism, however, is an important open research question. It is not solely determined by purely engineering questions, but also by the prospectives of the researchers.

Another research approach that emphasizes memory at the core of computing paradigm is the concept of "memcomputing" [95], [96]. Memcomputing goes further on the path of fusing computing and memory, with memory devices used as the active computing elements instead of transistors. It has been shown that non-Turing type memcomputing machines could solve some nondeterministic polynomial (NP)-complete problems in polynomial time, though also requiring a polynomial growth of memory [96]. Memcomputing therefore allows higher computational power than our proposed approach. However, it does not appear to feature intrinsic resilience to memory device imperfections. Nonetheless, it constitutes a powerful concept to which approaches integrating computing and memory should be confronted.

The present work leaves many open questions, at the technological level, but also at the computational level. We have shown that bioinspired programming may be connected to machine learning techniques. Nevertheless, the tasks presented in this work are still relatively simple textbook applications. More interdisciplinary research is needed to push the idea further and investigate the possibility of state-of-the-art machine learning. For this, we need to convince machine learning specialists to perform the needed specific research based on an understanding of the benefits but also of the limitations of nanodevices.

VII. Conclusion

In this work, we have explored a bioinspired methodology for programming nanodevices, which naturally implements an inference engine. The approach is fundamentally different from those of traditional electronics. Its source of inspiration is the Spike Timing Dependent Plasticity model of synaptic learning in neurosciences. The system puts memory at the very core of computing, with the physics of the memory devices playing an active role in the process of learning and inference. The system is capable of learning using an unsupervised paradigm. More precisely, it can identify the latent variable underlying the input to which it is exposed, and at the same time infer its classification among the latent variable values. The system is incredibly tolerant to device variations and read disturb issues.

We have shown that it may be adapted to various emergent memory device technologies. The most natural implementation uses cumulative memristive or phase change memory devices, in an analog manner. Another implementation can use stochastic devices like STT-MTJs. In many situations, stochastic and analog devices are equivalent. However, for most problems, this requires adding redundancy in the stochastic version.

We have explored several applications for the inference engine: image classification and detection of patterns in video and auditory data. These applications are impressive, especially given that they use precisely the same system. However, they are currently far from production and need to be associated with other circuits capable of exploiting the inference results. The methodology introduced in this work provides a new paradigm for nanoelectronics. Bioinspired programming of memory devices could identify latent variables and perform inference, which is the biggest challenge for processing Big Data. A simpler system can then interpret the inference result.

We have already connected the inference engine with Bayesian inference and the algorithm of Expectation-Maximization. By advancing this comparison further, it should be possible to develop significantly more complex and comprehensive inference engines, while retaining the vision of this work: using the bioinspired idea of fusing computing and memory, we intend to put inference at the core of nanoelectronics.

Appendix A
Derivation of the Expression of w_∞ in the Case of Cumulative Memristive Synapses

This appendix derives equation 6 from section IV. At the end of learning, if a synapse has reached a stable state, it experiences as many depression events as potentiation events. With the notations of section IV, this reads:

$$\delta w_+ p(PRE|POST) = \delta w_- p(\overline{PRE}|POST), \tag{9}$$

where we have introduced $p(\overline{PRE}|POST) = 1 - p(PRE|POST)$. By introducing the expressions of δw_+ and δw_- from equation 5, this becomes:

$$\begin{aligned}\alpha_+ \exp(-\beta_+ w_\infty) p(PRE|POST) = \\ \alpha_- \exp(-\beta_-(1-w_\infty))(1-p(PRE|POST)).\end{aligned} \tag{10}$$

Therefore, we have

$$\exp((\beta_+ + \beta_-)w_\infty - \beta_-) = \frac{\alpha_+}{\alpha_-}\frac{p(PRE|POST)}{(1-p(PRE|POST))}, \tag{11}$$

which leads to equation 6:

$$\begin{aligned}w_\infty =& \frac{\beta_-}{\beta_+ + \beta_-} \\ &+ \frac{1}{\beta_+ + \beta_-}\log\frac{p(PRE|POST)}{1-p(PRE|POST)} \\ &+ \frac{1}{\beta_+ + \beta_-}\log\frac{\alpha_+}{\alpha_-}.\end{aligned} \tag{12}$$

Appendix B
Derivation of the Expression of w_∞ in the Case of Stochastic Synapses

This appendix derives equation 8 from section IV. At the end of learning, if a synapse has reached a stable state, it experiences as many depression events as potentiation events. With the notations of section IV, and by introducing $p(State = 1)$ and $p(State = 0)$, the probabilities of the synapse to be in the 1 and 0 states, this reads:

$$\begin{aligned}p_+ \cdot p(PRE|POST) \cdot p(State = 0) = \\ p_- \cdot p(\overline{PRE}|POST) \cdot p(State = 1)\end{aligned} \tag{13}$$

With the notations of section IV, $p(State = 1) = w_\infty$ and $p(State = 0) = 1 - w_\infty$. If we introduce $\alpha = p_+/p_-$, equation 13 becomes

$$\alpha p(PRE|POST)(1 - w_\infty) = (1 - p(PRE|POST))w_\infty, \tag{14}$$

which leads to equation 8:

$$w_\infty = \frac{\alpha p(PRE|POST)}{1 + p(PRE|POST)(\alpha - 1)}. \tag{15}$$

Acknowledgments

The authors would like to thank C. Bennett, P. Bessière, L. Calvet, D. Chabi, D. Colliaux, B. De Salvo, J. Droulez, J. S. Friedman, J. Grollier, J.-O. Klein, J. Larroque, N. Locatelli, E. Mazer, A. Mizrahi, M. Suri, S. Tiwari, D. Vodenicarevic and W. S. Zhao.

References

[1] Y.-K. Chen, J. Chhugani, P. Dubey, C. Hughes, D. Kim, S. Kumar, V. Lee, A. Nguyen, and M. Smelyanskiy, "Convergence of Recognition, Mining, and Synthesis Workloads and Its Implications," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 790–807, May 2008.

[2] Q. V. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, and A. Y. Ng, "Building high-level features using large scale unsupervised learning," *International Conference on Machine Learning*, 2012.

[3] G. W. Burr, B. N. Kurdi, J. C. Scott, C. H. Lam, K. Gopalakrishnan, and R. S. Shenoy, "Overview of candidate device technologies for storage-class memory," *IBM J. Res. Dev.*, 2008.

[4] 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.

[5] 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, Apr. 2010.

[6] Z. Diao, Z. Li, S. Wang, Y. Ding, A. Panchula, E. Chen, L.-C. Wang, and Y. Huai, "Spin-transfer torque switching in magnetic tunnel junctions and spin-transfer torque random access memory," *Journal of Physics: Condensed Matter*, vol. 19, no. 16, p. 165209, Apr. 2007.

[7] S. Yu, X. Guan, and H.-S. Wong, "On the stochastic nature of resistive switching in metal oxide RRAM: Physical modeling, monte carlo simulation, and experimental characterization," in *IEDM Tech. Dig.*, Dec. 2011, pp. 17.3.1 –17.3.4.

[8] T. Devolder, J. Hayakawa, K. Ito, H. Takahashi, S. Ikeda, P. Crozat, N. Zerounian, J.-V. Kim, C. Chappert, and H. Ohno, "Single-Shot Time-Resolved Measurements of Nanosecond-Scale Spin-Transfer Induced Switching: Stochastic Versus Deterministic Aspects," *Phys. Rev. Lett.*, vol. 100, no. 5, p. 057206, Feb. 2008.

[9] E. Marder and J.-M. Goaillard, "Variability, compensation and homeostasis in neuron and network function," *Nat Rev Neurosci*, vol. 7, no. 7, pp. 563–574, Jul. 2006.

[10] Y. V. Pershin, S. La Fontaine, and M. Di Ventra, "Memristive model of amoeba learning," *Phys. Rev. E*, vol. 80, no. 2, p. 021926, 2009.

[11] B. Linares-Barranco and T. Serrano-Gotarredona, "Exploiting memristance in adaptive asynchronous spiking neuromorphic nanotechnology systems," in *Proc. of IEEE Conference on Nanotechnology, 2009*, 2009, pp. 601–604.

[12] S. H. Jo, K.-H. Kim, and W. Lu, "Programmable Resistance Switching in Nanoscale Two-Terminal Devices," *Nano Lett.*, vol. 9, no. 1, pp. 496–500, Jan. 2009.

[13] G. S. Snider, "Self-organized computation with unreliable, memristive nanodevices," *Nanotechnology*, vol. 18, no. 36, p. 365202, Sep. 2007.

[14] K. Seo, I. Kim, S. Jung, M. Jo, S. Park, J. Park, J. Shin, K. P. Biju, J. Kong, K. Lee, B. Lee, and H. Hwang, "Analog memory and spike-timing-dependent plasticity characteristics of a nanoscale titanium oxide bilayer resistive switching device," *Nanotechnology*, vol. 22, no. 25, p. 254023, Jun. 2011.

[15] M. Versace and B. Chandler, "The brain of a new machine," *Spectrum, IEEE*, vol. 47, no. 12, pp. 30–37, 2010.

[16] D. Querlioz, O. Bichler, and C. Gamrat, "Simulation of a memristor-based spiking neural network immune to device variations," *Proc. of the Int. Joint Conf. on Neural Networks (IJCNN)*, pp. 1775 – 1781, 2011.

[17] M. Suri, O. Bichler, D. Querlioz, O. Cueto, L. Perniola, V. Sousa, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Phase change memory as synapse for ultra-dense neuromorphic systems: Application to complex visual pattern extraction," in *IEDM Tech. Dig.* IEEE, Dec. 2011, pp. 4.4.1–4.4.4.

[18] S. Yu, Y. Wu, R. Jeyasingh, D. Kuzum, and H. P. Wong, "An Electronic Synapse Device Based on Metal Oxide Resistive Switching Memory for Neuromorphic Computation," *IEEE Trans. Electron Dev.*, vol. 58, no. 8, pp. 2729–2737, Aug. 2011.

[19] V. Erokhin, T. Berzina, P. Camorani, A. Smerieri, D. Vavoulis, J. Feng, and M. P. Fontana, "Material Memristive Device Circuits with Synaptic Plasticity: Learning and Memory," *BioNanoScience*, vol. 1, no. 1-2, pp. 24–30, Apr. 2011.

[20] 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," *Nat. Mat.*, vol. 11, no. 10, pp. 860–864, 2012.

[21] 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.

[22] M. Sharad, C. Augustine, G. Panagopoulos, and K. Roy, "Spin-Based Neuron Model With Domain-Wall Magnets as Synapse," *IEEE Transactions on Nanotechnology*, vol. 11, no. 4, pp. 843 –853, Jul. 2012.

[23] J. H. Lee and K. K. Likharev, "Defect-tolerant nanoelectronic pattern classifiers," *Int. J. Circ. Theor. Appl.*, vol. 35, no. 3, pp. 239–264, May 2007.

[24] K. Cantley, A. Subramaniam, H. Stiegler, R. Chapman, and E. Vogel, "Hebbian Learning in Spiking Neural Networks With Nanocrystalline Silicon TFTs and Memristive Synapses," *IEEE Transactions on Nanotechnol.*, vol. 10, no. 5, pp. 1066 –1073, Sep. 2011.

[25] D. Chabi, D. Querlioz, W. Zhao, and J.-O. Klein, "Robust Learning Approach for Neuro-inspired Nanoscale Crossbar Architecture," *J. Emerg. Technol. Comput. Syst.*, vol. 10, no. 1, pp. 5:1–5:20, Jan. 2014.

[26] K. Gacem, J.-M. Retrouvey, D. Chabi, A. Filoramo, W. Zhao, J.-O. Klein, and V. Derycke, "Neuromorphic function learning with carbon nanotube based synapses," *Nanotechnology*, vol. 24, no. 38, p. 384013, Sep. 2013.

[27] S.-Y. Liao, J.-M. Retrouvey, G. Agnus, W. Zhao, C. Maneux, S. Fregonese, T. Zimmer, D. Chabi, A. Filoramo, V. Derycke, C. Gamrat, and J.-O. Klein, "Design and Modeling of a Neuro-Inspired Learning Circuit Using Nanotube-Based Memory Devices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 9, pp. 2172–2181, Sep. 2011.

[28] O. Kavehei, "Highly Scalable Neuromorphic Hardware with 1-bit Stochastic nano-Synapses," arXiv e-print 1309.6419, Sep. 2013.

[29] S. Saïghi, C. G. Mayr, T. Serrano-Gotarredona, H. Schmidt, G. Lecerf, J. Tomas, J. Grollier, S. Boyn, A. F. Vincent, D. Querlioz, S. La Barbera, F. Alibart, D. Vuillaume, O. Bichler, C. Gamrat, and B. Linares-Barranco, "Plasticity in memristive devices for spiking neural networks," *Front. Neurosci*, vol. 9, p. 51, 2015.

[30] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Häfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, S. Saighi, J. Wijekoon, and K. Boahen, "Neuromorphic silicon neuron circuits," *Front. Neuromorphic Engineering*, vol. 5, p. 73, 2011.

[31] B. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. Arthur, P. 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.

[32] 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.

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

[34] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, Jun. 2010, pp. 1947 –1950.

[35] C. Mead, *Analog VLSI and Neural Systems*, 1st ed. Addison Wesley Publishing Company, Jan. 1989.

[36] B. Nessler, M. Pfeiffer, L. Buesing, and W. Maass, "Bayesian Computation Emerges in Generic Cortical Microcircuits through Spike-Timing-Dependent Plasticity," *PLoS Comput Biol*, vol. 9, no. 4, Apr. 2013.

[37] G.-Q. Bi and M.-M. Poo, "Synaptic modification by correlated activity: Hebb's Postulate Revisited," *Annu. Rev. Neurosci.*, vol. 24, no. 1, pp. 139–166, Mar. 2001.

[38] A. Perfors, J. B. Tenenbaum, T. L. Griffiths, and F. Xu, "A tutorial introduction to Bayesian models of cognitive development," *Cognition*, vol. 120, no. 3, pp. 302–321, Sep. 2011.

[39] H. Markram, J. Lubke, M. Frotscher, and B. Sakmann, "Regulation of Synaptic Efficacy by Coincidence of Postsynaptic APs and EPSPs," *Science*, vol. 275, no. 5297, pp. 213–215, Jan. 1997.

[40] L. F. Abbott and S. B. Nelson, "Synaptic plasticity: taming the beast," *Nat Neurosci*, vol. 3, pp. 1178–1183, 2000.

[41] P. J. Sjöström, E. A. Rancz, A. Roth, and M. Häusser, "Dendritic Excitability and Synaptic Plasticity," *Physiological Reviews*, vol. 88, no. 2, pp. 769–840, Apr. 2008.

[42] J. M. Brader, W. Senn, and S. Fusi, "Learning Real-World Stimuli in a Neural Network with Spike-Driven Synaptic Dynamics," *Neural Comput.*, vol. 19, no. 11, pp. 2881–2912, Nov. 2007.

[43] 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.

[44] T. Masquelier, R. Guyonneau, and S. J. Thorpe, "Spike Timing Dependent Plasticity Finds the Start of Repeating Patterns in Continuous Spike Trains," *PLoS ONE*, vol. 3, no. 1, p. e1377, Jan. 2008.

[45] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[46] 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. 2278–2324, 1998.

[47] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, High Performance Convolutional Neural Networks for Image Classification," in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, ser. IJCAI'11. Barcelona, Catalonia, Spain: AAAI Press, 2011, pp. 1237–1242.

[48] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, 2012.

[49] J. V. Arthur and K. A. Boahen, "Learning in silicon: Timing is everything," *Advances in neural information processing systems*, vol. 18, pp. 281–1185, 2006.

[50] G. Indiveri, E. Chicca, and R. Douglas, "A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 211–221, 2006.

[51] M. Azghadi, N. Iannella, S. Al-Sarawi, G. Indiveri, and D. Abbott, "Spike-Based Synaptic Plasticity in Silicon: Design, Implementation, Application, and Challenges," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 717–737, May 2014.

[52] G. Snider, "Spike-timing-dependent learning in memristive nanodevices," in *Prof. of IEEE International Symposium on Nanoscale Architectures 2008 (NANOARCH)*, 2008, pp. 85–92.

[53] 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, 2012.

[54] A. Afifi, A. Ayatollahi, and F. Raissi, "Implementation of biologically plausible spiking neural network models on the memristor crossbar-based CMOS/nano circuits," in *European Conference on Circuit Theory and Design (ECCTD)*, 2009, pp. 563–566.

[55] G. Lecerf, J. Tomas, and S. Saighi, "Excitatory and Inhibitory Memristive Synapses for Spiking Neural Networks," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2013, pp. 1616–1619.

[56] J. V. Arthur and K. A. Boahen, "Silicon-Neuron Design: A Dynamical Systems Approach," *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 58, no. 5, pp. 1034–1043, 2011.

[57] D. Querlioz, P. Dollfus, O. Bichler, and C. Gamrat, "Learning with memristive devices: how should we model their behavior?" *Proc. of IEEE/ACM Int. Symp. Nanoscale Architectures (NANOARCH 2011)*, p. 150, 2011.

[58] M. Suri, O. Bichler, D. Querlioz, B. Traoré, O. Cueto, L. Perniola, V. Sousa, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Physical aspects of low power synapses based on phase change memory devices," *Journal of Applied Physics*, vol. 112, no. 5, pp. 054 904–054 904–10, Sep. 2012.

[59] 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, 2013.

[60] A. Vincent, N. Locatelli, J.-O. Klein, W. Zhao, S. Galdin-Retailleau, and D. Querlioz, "Analytical Macrospin Modeling of the Stochastic Switching Time of Spin-Transfer Torque Devices," *IEEE Transactions on Electron Devices*, vol. 62, no. 1, pp. 164–170, Jan. 2015.

[61] F. Alibart, E. Zamanidoost, and D. B. Strukov, "Pattern classification by memristive crossbar circuits using ex situ and in situ training," *Nat Commun*, vol. 4, Jun. 2013.

[62] A. Chanthbouala, R. Matsumoto, J. Grollier, V. Cros, A. Anane, A. Fert, A. V. Khvalkovskiy, K. A. Zvezdin, K. Nishimura, Y. Nagamine, H. Maehara, K. Tsunekawa, A. Fukushima, and S. Yuasa, "Vertical-current-induced domain-wall motion in MgO-based magnetic tunnel junctions with low current densities," *Nat. Phys.*, vol. 7, no. 8, pp. 626–630, 2011.

[63] D. Querlioz, O. Bichler, P. Dollfus, and C. Gamrat, "Immunity to Device Variations in a Spiking Neural Network with Memristive Nanodevices," *IEEE Trans. Nanotechnol.*, vol. 12, no. 3, pp. 288 – 295, 2013.

[64] G. Lecerf, J. Tomas, S. Boyn, S. Girod, A. Mangalore, J. Grollier, and S. Saighi, "Silicon neuron dedicated to memristive spiking neural networks," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, Jun. 2014, pp. 1568–1571.

[65] O. Bichler, M. Suri, D. Querlioz, D. Vuillaume, B. DeSalvo, and C. Gamrat, "Visual Pattern Extraction Using Energy-Efficient "2-PCM Synapse" Neuromorphic Architecture," *IEEE Trans. Electron Devices*, vol. 59, no. 8, pp. 2206 – 2214, 2012.

[66] M. Suri, D. Garbin, O. Bichler, D. Querlioz, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Impact of PCM resistance-drift in neuromorphic systems and drift-mitigation strategy," in *2013 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, Jul. 2013, pp. 140–145.

[67] M. Suri, O. Bichler, D. Querlioz, G. Palma, E. Vianello, D. Vuillaume, C. Gamrat, and B. DeSalvo, "CBRAM Devices as Binary Synapses for Low-Power Stochastic Neuromorphic Systems: Auditory (Cochlea) and Visual (Retina) Cognitive Processing Applications," *IEDM Tech. Dig.*, p. 10.3.1, 2012.

[68] W. Senn and S. Fusi, "Convergence of stochastic learning in perceptrons with binary synapses," *Phys. Rev. E*, vol. 71, no. 6, p. 061907, Jun. 2005.

[69] Y. Kondo and Y. Sawada, "Functional abilities of a stochastic logic neural network," *IEEE Transactions on Neural Networks*, vol. 3, no. 3, pp. 434 –443, May 1992.

[70] Y. Zhang, W. Zhao, J.-O. Klein, W. Kang, D. Querlioz, Y. Zhang, D. Ravelosona, and C. Chappert, "Spintronics for low-power computing," in *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, Mar. 2014, pp. 1–6.

[71] A. Vincent, J. Larroque, N. Locatelli, N. Ben Romdhane, O. Bichler, C. Gamrat, W. Zhao, J.-O. Klein, S. Galdin-Retailleau, and D. Querlioz, "Spin-Transfer Torque Magnetic Memory as a Stochastic Memristive Synapse for Neuromorphic Systems," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 9, no. 2, pp. 166–174, Apr. 2015.

[72] S. Gaba, P. Sheridan, J. Zhou, S. Choi, and W. Lu, "Stochastic memristive devices for computing and neuromorphic applications," *Nanoscale*, vol. 5, no. 13, pp. 5872–5878, Jun. 2013.

[73] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, and H.-S. P. Wong, "Stochastic learning in oxide binary synaptic device for neuromorphic computing," *Front Neurosci*, vol. 7, Oct. 2013.

[74] Y. Zhang, W. Zhao, G. Prenat, T. Devolder, J.-O. Klein, C. Chappert, B. Dieny, and D. Ravelosona, "Electrical Modeling of Stochastic Spin Transfer Torque Writing in Magnetic Tunnel Junctions for Memory and Logic Applications," *IEEE Transactions on Magnetics*, vol. 49, no. 7, pp. 4375–4378, Jul. 2013.

[75] A. Fukushima, T. Seki, K. Yakushiji, H. Kubota, H. Imamura, S. Yuasa, and K. Ando, "Spin dice: A scalable truly random number generator based on spintronics," *Appl. Phys. Express*, vol. 7, no. 8, p. 083001, Aug. 2014.

[76] J. Joshua Yang, M. X. Zhang, M. D. Pickett, F. Miao, J. Paul Strachan, W.-D. Li, W. Yi, D. A. A. Ohlberg, B. Joon Choi, W. Wu, J. H. Nickel, G. Medeiros-Ribeiro, and R. Stanley Williams, "Engineering nonlinearity into memristors for passive crossbar applications," *Appl. Phys. Lett.*, vol. 100, no. 11, pp. 113 501–113 501–4, Mar. 2012.

[77] 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, 2012.

[78] 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.

[79] D. Querlioz, W. S. Zhao, P. Dollfus, J.-O. Klein, O. Bichler, and C. Gamrat, “Bioinspired Networks with Nanoscale Memristive Devices that Combine the Unsupervised and Supervised Learning Approaches,” in *Proc. of NANOARCH*, 2012.

[80] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, “Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition,” *CoRR abs/1003.0358*, Mar. 2010.

[81] P. Lichtsteiner, C. Posch, and T. Delbruck, “A 128x 128 120 dB 15 mus Latency Asynchronous Temporal Contrast Vision Sensor,” *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[82] http://sourceforge.net/p/jaer/wiki/AER data/.

[83] A. F. Vincent, J. Larroque, W. S. Zhao, N. Ben Romdhane, O. Bichler, C. Gamrat, J.-O. Klein, S. Galdin-Retailleau, and D. Querlioz, “Spin-transfer torque magnetic memory as a stochastic memristive synapse,” in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, Jun. 2014, pp. 1074–1077.

[84] V. Chan, S.-C. Liu, and A. van Schaik, “AER EAR: A Matched Silicon Cochlea Pair With Address Event Representation Interface,” *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 54, no. 1, pp. 48–59, 2007.

[85] W. Maass, “Noise as a Resource for Computation and Learning in Networks of Spiking Neurons,” *Proceedings of the IEEE*, vol. 102, no. 5, pp. 860–880, May 2014.

[86] A. Dempster, N. Laird, and D. Rubin, “Maximum Likelihood from Incomplete Data via the EM Algorithm,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[87] R. Beach, T. Min, C. Horng, Q. Chen, P. Sherman, S. Le, S. Young, K. Yang, H. Yu, X. Lu, W. Kula, T. Zhong, R. Xiao, A. Zhong, G. Liu, J. Kan, J. Yuan, J. Chen, R. Tong, J. Chien, T. Torng, D. Tang, P. Wang, M. Chen, S. Assefa, M. Qazi, J. DeBrosse, M. Gaidis, S. Kanakasabapathy, Y. Lu, J. Nowak, E. O’Sullivan, T. Maffitt, J. Sun, and W. Gallagher, “A statistical study of magnetic tunnel junctions for high-density spin torque transfer-MRAM (STT-MRAM),” in *Electron Devices Meeting, 2008. IEDM 2008. IEEE International*, Dec. 2008, pp. 1–4.

[88] D. Worledge, G. Hu, P. Trouilloud, D. Abraham, S. Brown, M. Gaidis, J. Nowak, E. O’Sullivan, R. Robertazzi, J. Sun, and W. Gallagher, “Switching distributions and write reliability of perpendicular spin torque MRAM,” in *Electron Devices Meeting (IEDM), 2010 IEEE International*, Dec. 2010, pp. 12.5.1–12.5.4.

[89] C. Bartolozzi, O. Nikolayeva, and G. Indiveri, “Implementing homeostatic plasticity in VLSI networks of spiking neurons,” in *IEEE Int. Conf. Electronics, Circuits and Systems (ICECS 2008)*, 2008, pp. 682–685.

[90] H. S. Stone, “A Logic-in-Memory Computer,” *IEEE Transactions on Computers*, vol. C-19, no. 1, pp. 73–78, Jan. 1970.

[91] S. Matsunaga, J. Hayakawa, S. Ikeda, K. Miura, H. Hasegawa, T. Endoh, H. Ohno, and T. Hanyu, “Fabrication of a Nonvolatile Full Adder Based on Logic-in-Memory Architecture Using Magnetic Tunnel Junctions,” *Appl. Phys. Express*, vol. 1, no. 9, p. 091301, Sep. 2008.

[92] W. Zhao, M. Moreau, E. Deng, Y. Zhang, J.-M. Portal, J.-O. Klein, M. Bocquet, H. Aziza, D. Deleruyelle, C. Muller, D. Querlioz, N. Ben Romdhane, D. Ravelosona, and C. Chappert, “Synchronous Non-Volatile Logic Gate Design Based on Resistive Switching Memories,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 2, pp. 443–454, Feb. 2014.

[93] N. Locatelli, A. F. Vincent, A. Mizrahi, J. S. Friedman, D. Vodenicarevic, J.-V. Kim, J.-O. Klein, W. Zhao, J. Grollier, and D. Querlioz, “Spintronic devices as key elements for energy-efficient neuroinspired architectures,” in *Design, Automation Test in Europe Conference Exhibition (DATE), 2015*, Mar. 2015, pp. 994–999.

[94] M. Natsui, D. Suzuki, N. Sakimura, R. Nebashi, Y. Tsuji, A. Morioka, T. Sugibayashi, S. Miura, H. Honjo, K. Kinoshita, S. Ikeda, T. Endoh, H. Ohno, and T. Hanyu, “Nonvolatile logic-in-memory array processor in 90nm MTJ/MOS achieving 75% leakage reduction using cycle-based power gating,” in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2013 IEEE International*, Feb. 2013, pp. 194–195.

[95] Y. Pershin and M. Di Ventra, “Neuromorphic, Digital, and Quantum Computation With Memory Circuit Elements,” *Proceedings of the IEEE*, vol. 100, no. 6, pp. 2071 –2080, Jun. 2012.

[96] F. Traversa and M. Di Ventra, “Universal Memcomputing Machines,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–1, 2015.

PLACE PHOTO HERE

Damien Querlioz received the M. S. degree from Ecole Normale Superieure, Paris in 2005 and the Ph.D. degree from the Univ. Paris-Sud, France, in 2008. After postdocs at Stanford University and at CEA LIST, he became a CNRS research scientist with Univ. Paris-Sud in 2010. He develops new concepts in nanoelectronics and spintronics relying on bio-inspiration. His research interests have also included the physics of advanced nanodevices. He leads the ANR CogniSpin project, which investigates the use of magnetic memory as synapses. He leads the CNRS/MI DEFIBAYES project and is a one of the lead PI of the FP7 FETOPEN BAMBI project, which explore the new paradigms for nanolectronics based on Bayesian inference.

PLACE PHOTO HERE

Olivier Bichler received the M.S. degree in embedded systems from the Ecole Normale Superieure de Cachan, France, in 2009 and the Ph. D. degree from the Universite Paris-Sud, Orsay, France, in 2012. He is now a Research Engineer at CEA LIST, France, and develops novel architectures based on nanoelectronics and bio-inspired neuromorphic computing.

PLACE PHOTO HERE

Adrien Francis Vincent received the M.S. degree from the Ecole Normale Superieure de Cachan, France, in 2013. During his Ph.D. at Univ. Paris-Sud, he is studying the integration of spintronic nanodevices in neuromorphic architectures.

PLACE PHOTO HERE

Christian Gamrat received a degree in electrical engineering from the Université Joseph Fourier, Grenoble, France, in 1979 and a degree in information processing in 1993 from Ecole Nationale Supérieure dElectronique et de Radioélectricité, Grenoble, France. In 1981, he started his career at CEA/DSM Grenoble on the design of high speed data acquisition systems for solid state and nuclear physics experiments, got involved in the study and design of neural networks computing machines in 1987, and led the team for the MIND-1024 neurocomputer project in 1989. In 1994, he joined the Parallel Computing Architecture Lab of CEA near Paris, where he finalized the development of the SYMPHONIE embedded massively parallel computer for use on board military fighter aircraft. In 1997 he started activity on hardware reconfigurable computing, and in 2003, he initiated research on novel computing architectures aimed at nanotechnologies. He is currently a Senior Expert in the field of advanced computing architectures and nanocomputing, and he leads the Nanocomputing group with CEA LIST, Gif-sur-Yvette, France.