ADVANCED INTELLIGENT SYSTEMS
Open Access
www.advintellsyst.com

Memristors—From In-Memory Computing, Deep Learning Acceleration, and Spiking Neural Networks to the Future of Neuromorphic and Bio-Inspired Computing

Adnan Mehonic, Abu Sebastian, Bipin Rajendran, Osvaldo Simeone, Eleni Vasilaki, and Anthony J. Kenyon*

Machine learning, particularly in the form of deep learning (DL), has driven most of the recent fundamental developments in artificial intelligence (AI). DL is based on computational models that are, to a certain extent, bio-inspired, as they rely on networks of connected simple computing units operating in parallel. The success of DL is supported by three factors: availability of vast amounts of data, continuous growth in computing power, and algorithmic innovations. The approaching demise of Moore's law, and the consequent expected modest improvements in computing power that can be achieved by scaling, raises the question of whether the progress will be slowed or halted due to hardware limitations. This article reviews the case for a novel beyond-complementary metal–oxide–semiconductor (CMOS) technology—memristors—as a potential solution for the implementation of power-efficient in-memory computing, DL accelerators, and spiking neural networks. Central themes are the reliance on non-von-Neumann computing architectures and the need for developing tailored learning and inference algorithms. To argue that lessons from biology can be useful in providing directions for further progress in AI, an example-based reservoir computing is briefly discussed. At the end, speculation is given on the "big picture" view of future neuromorphic and brain-inspired computing systems.

1. Introduction

Three factors are currently driving the main developments in artificial intelligence (AI): availability of vast amounts of data, continuous growth in computing power, and algorithmic innovations. Graphics processing units (GPUs) have been demonstrated as effective co-processors for the implementation of machine learning (ML) algorithms based on deep learning (DL). Solutions based on DL and GPU implementations have led to massive improvements in many AI tasks, but have also caused an exponential increase in demand for computing power. Recent analyses show that the demand for computing power has increased by a factor of 300 000 since 2012, and the estimate is that this demand will double every 3.4 months—at a much faster rate than improvements made historically through Moore's scaling (a sevenfold improvement over the same period of time).[1] At the same time, Moore's law has been slowing down significantly for the last few years,[2] as there are strong indications that we will not be able to continue scaling down complementary metal–oxide–semiconductor (CMOS) transistors. This calls for the exploration of alternative technology roadmaps for the development of scalable and efficient AI solutions.

Transistor scaling is not the only way to improve computing performance. Architectural innovations, such as GPUs, field-programmable arrays (FPGAs), and application-specific integrated circuits (ASICs), have all significantly advanced the ML field.[3] A common aspect of modern computing architectures for ML is a move away from the classical von-Neumann architecture that physically separates memory and computing. This approach yields a performance bottleneck that is often the main reason for both energy and speed inefficiency of ML implementations on conventional hardware platforms due to

Dr. A. Mehonic, Prof. A. J. Kenyon
Department of Electronic & Electrical Engineering
UCL
Torrington Place, London WC1E 7JE, UK
E-mail: adnan.mehonic.09@ucl.ac.uk

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/aisy.202000085.

DOI: 10.1002/aisy.202000085

Dr. A. Sebastian
IBM Research Europe
Rüschlikon, Zurich 8803, Switzerland

Dr. B. Rajendran, Prof. O. Simeone
Department of Engineering
King's College London
London WC2R 2LS, UK

Prof. E. Vasilaki
Department of Computer Science
University of Sheffield
Regent Court (DCS) 211 Portobello, Sheffield S1 4DP, South Yorkshire, UK

costly data movements. However, architectural developments alone are not likely to be sufficient. In fact, standard digital CMOS components are inherently not well suited for the implementation of a massive number of continuous weights/synapses in artificial neural networks (ANNs).

1.1. The Promise of Memristors

There is a strong case to be made for the exploration of alternative technologies. Although the memristor technology is currently still in development, it is a strong candidate for future non-CMOS and beyond von-Neumann computing solutions.[4] Since its early development in 2008,[5] or even earlier under different names,[6] memristor technology expanded remarkably to include many different materials solutions, physical mechanisms, and novel computing approaches.[4] A single progress report cannot cover all different approaches and fast-growing developments in the field. The evaluation of state of the art in memristor-based electronics can be found elsewhere.[7] Instead, in this article, we present and discuss a few representative case studies, showcasing the potential role of memristors in the expanding field of AI hardware. We present the examples of how memristors are used for in-memory computing systems, DL accelerators, and spike-based computing. Finally, we discuss and speculate on the future of neuromorphic and bio-inspired computing paradigms and provide reservoir computing as an example.

For the last 15 years, memristors have been a focal point for many different research communities—mathematicians, solid-state physicists, experimental material scientists, electrical engineers, and, more recently, computer scientists and computational neuroscientists. The concept of memristor was introduced almost 50 years ago, back in 1971,[8] was nearly forgotten for almost four decades. There are many different flavors of memristive technologies. Still, in their most popular implementation, memristors are simple two-terminal devices with the extraordinary property that their resistance depends on their history of electrical stimuli. In other words, memristors are resistors with memory. They promise high levels of integration, stable non-volatile resistance states, fast resistance switching, and excellent energy efficiency—all very desirable properties for next generation of memory technologies.

The physical implementations of memristors are broad and arguably include many different technologies, such as redox-based resistive random-access memory (ReRAM), phase-change memories (PCMs), and magnetoresistive random-access memory (MRAM). Further differentiations within larger classes can be made, depending on physical mechanisms that govern the resistance change. Many excellent reviews cover the principles and switching mechanisms of memristor devices. Here, we will briefly mention two extensively studied types of memristive devices, namely, ReRAM and PCM.

Resistance switching is one of the most explored properties of memristive devices. A thin insulating film reversibly changes its electrical resistance—between an insulating state and a conducting state—under the application of an external electrical stimulus. For binary memory devices, two stable states are sought, typically called the high resistance state (HRS) and the low resistance state (LRS). The transition from the HRS to the LRS is called a SET process, whereas a RESET process describes the transition from the LRS to the HRS.

Basic memory cells of both types, in their most straightforward implementation, have three layers—two conductive electrodes and a thin switching layer sandwiched in-between. Local redox processes govern resistance switching in ReRAM devices. A broad classification can be made based on a distinction between the switching that happens as a result of intrinsic properties of the switching material (typically oxides) and switching that is the result of in-diffusion of metal ions (typically from one of the metallic electrodes). The former type is called intrinsic switching, and the latter is called extrinsic switching.[9] Alternatively, a classification can be made depending on the main driving force for the redox process (thermal or electrical), or the type of ions that move. The main three classes are electrochemical metallization cells (or conductive bridge) ReRAMs (ECM), valence change ReRAMs (VCM), and thermochemical ReRAMs (TCM).[4]

Many ReRAM devices require an electroforming step prior to resistance switching. This can be considered a soft breakdown of the insulating material. A conductive filament is produced inside the insulating film as a result of the applied electrical bias. Modification of conductive filaments, led by a local redox process, leads to the change of resistance. The diameter of the conductive filament is typically on the order of a few nanometers to a few tens of nanometers, and it does not depend on the size of the electrodes. Another, less common type is interface-type switching, which does not depend on creation and modification of conductive filaments, but can be driven by the formation of a tunnel or Schottky barrier across the whole interface between electrode and switching layer.

In the case of PCMs, the change of resistance due to the crystallization and amorphization processes of phase change materials. Amplitude and duration of applied voltage pulses control the phase transitions – the SET process changes the amorphous to a crystalline phase (HRS to LRS transition), and the RESET process changes the crystalline to an amorphous phase (LRS to HRS transition).

For many computing tasks, more than two states are required, and for most memristive devices, including ReRAMs and PCMs, many resistance states can be achieved. However, benchmarking of memristive devices for different applications, beyond pure digital memory, can be challenging and relies on many different parameters other than the number of different resistance states. We will discuss the main device properties in the context of different applications.

1.2. The Landscape of Different Approaches and Applications

In the context of this article, memristors can be used in applications beyond simple memory devices.[10] A "big picture" landscape of memristor-based approaches for AI is shown in **Figure 1**. There is more than one way that memristors can perform computing. A unique feature of memristor devices is the ability to co-locate memory and computing and to break the von-Neumann bottleneck at the lowest, nanometer-scale level. One such approach is the concept of in-memory computing, which uses memory not only to store the data but also to perform computation at the same physical location. Furthermore, memristors

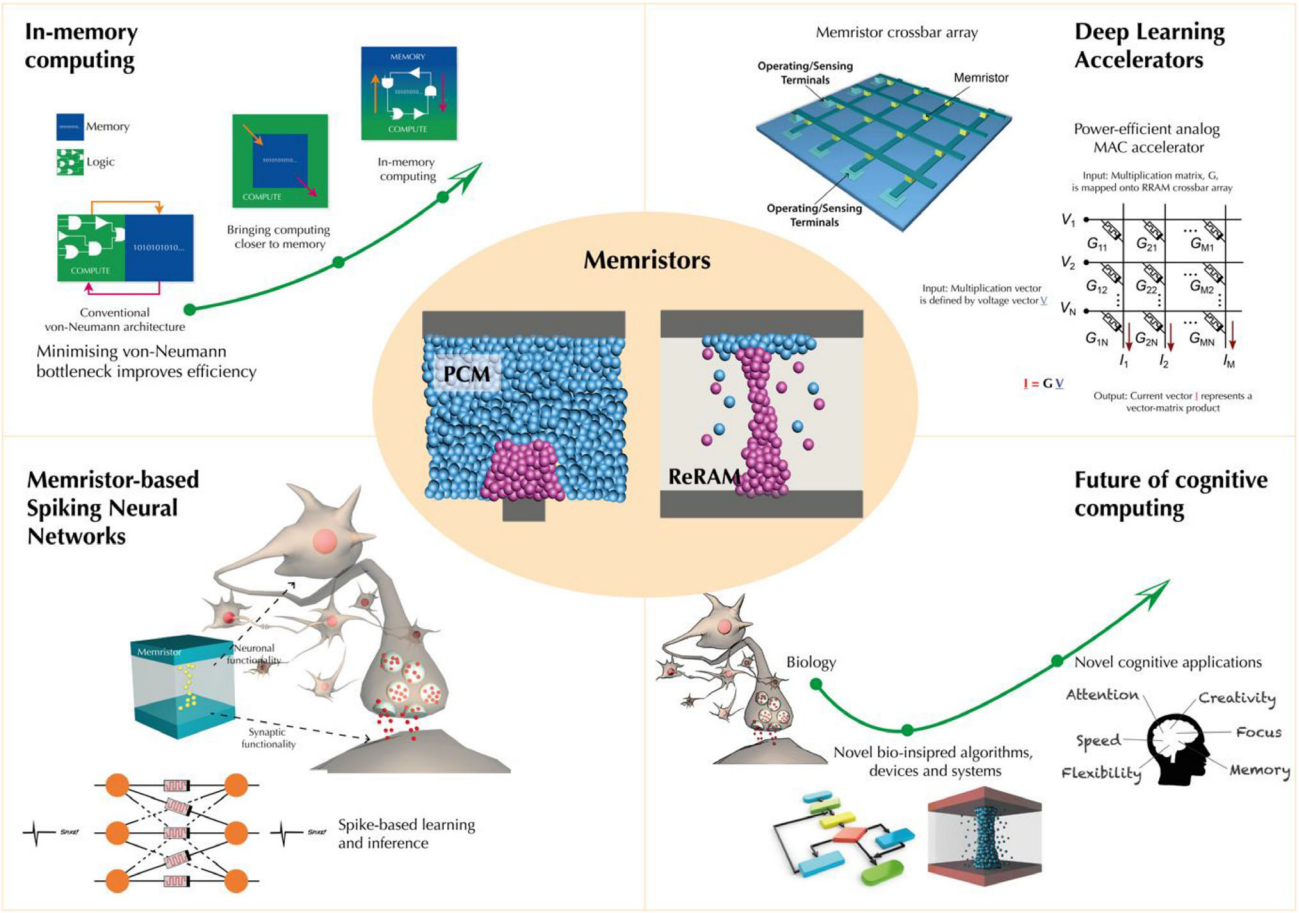


Figure 1. The landscape of memristor-based systems for AI. In-memory computing aims to eliminate the von-Neumann bottleneck by implementing compute directly within the memory. DL accelerators based on memristive crossbars are used to implement vector-matrix multiplication directly using Ohm's and Kirchhoff's laws. SNNs, a type of ANNs, are biologically more plausible and do not operate with continuous signals, but use spikes to process and transfer data. Memristor systems could provide a hardware platform to implement spike-based learning and inference. More complex functionalities (neuromorphic), beyond simple digital switching CMOS paradigm, directly implemented in memristive hardware primitives, might fuel the next wave of higher cognitive systems.

have long been considered for DL acceleration. In particular, memristive crossbar arrays physically represent weights in ANNs as conductances at each cross-point. When voltages are applied at one side of the crossbar and current sensed on the orthogonal terminals, the array provides vector-matrix multiplication in constant time step using Kirchhoff's and Ohm's laws. Vector-matrix multiplications dominate most DL algorithms—hundreds of thousands are often needed during training and inference. When weights are implemented as memristor conductances, there is no need for the extensive power-hungry data movement required by conventional digital systems based on the von-Neumann architecture.

Other more bio-realistic concepts are also being explored. These include schemes relying on spike-based communication. The central premise of this approach can be summarized with the motto "computing with time, not in time." It has been shown that memristors can directly implement some functions of biological neurons and synapses, most importantly, synapse-like plasticity, and neuron-like integration and spiking. In these solutions, the information is encoded and transferred in the form of voltage or current spikes. Memristor resistances are used as proxies for synaptic strengths. More importantly, adjustment of the resistances is controlled according to local learning rules. One popular local learning rule is spike-timing-dependent plasticity (STDP), which adjust a local state variable such as conductance dynamically based on the relative timing of spikes. In a simple example, the conductance of a memristive "synapse" can be increased or decreased depending on the degree of overlap between pre- and post-synaptic voltage pulses. There also exist implementations that do not require overlapping pulses, instead utilizing the volatile internal dynamics of memristive devices. Spike-based computing promises further improvements in power efficiency, taking the inspiration from the remarkable efficiency of the human brain. It is important to note that there exist many challenges related to full adoption of memristor technologies. We discuss some of the common device and system non-idealities and some proposed schemes to deal with these. In contrast, when carefully controlled, some of these

non-idealities, such as stochastic switching, can be harnessed for probabilistic computing. We provide an overview of some algorithmic approaches of probabilistic spiking neural networks (SNNs) and provide references to hardware and memristor-based implementations.

Finally, we speculate that, for future developments in AI, new knowledge and computational models from the fields of computational neuroscience could play a crucial role. Virtually, all recent developments in ML and DL are driven by the field of computer science. At the same time, the algorithmic inspiration from neuroscience is mostly based on old models established as early as the 1950s. Although we are still at the infancy of understanding the full working principles of the biological brain, novel brain-inspired architectural principles, beyond simple probabilistic DL approaches, could lead to higher level cognitive functionalities. One such example is the concept of reservoir computing, which we discuss briefly in the article. It is unlikely that current digital CMOS transistor technology can be optimized for the implementation of much more dynamic and adaptive systems in an efficient way. In contrast, memristor-based systems, with their rich switching dynamics and many state variables, may provide a perfect substrate to build a new class of intelligent and efficient neuromorphic systems.

2. In-Memory Computing

In the von-Neumann architecture, which dates back to the 1940s, memory and processing units are physically separated, and large amounts of data need to be shuttled back and forth between them during the execution of various computational tasks. The latency and energy associated with accessing data from the memory units are key performance bottlenecks for a range of applications, in particular, for the increasingly prominent AI-related workloads.[11] The energy cost associated with moving data is a key challenge for both severely energy constrained mobile and edge computing as well as high-performance computing in a cloud environment due to cooling constraints. The current approaches, such as using hundreds of processors in parallel[12] or application-specific processors,[13] are not likely to fully overcome the challenge of data movement. It is getting increasingly clear that novel architectures need to be explored where memory and processing are better collocated.

In-memory computing is one such non-von-Neumann approach where certain computational tasks are performed in place in the memory itself organized as a computational memory unit.[14–17] As schematically shown in **Figure 2**, in-memory computing obviates the need to move data into a processing unit. Computing is performed by exploiting the physical attributes of the memory devices, their array-level organization, the peripheral circuitry, and the control logic. In this paradigm, the memory is an active participant in the computational task. Besides reducing latency and energy cost associated with data movement, in-memory computing also has the potential to improve the computational time complexity associated with certain tasks due to the massive parallelism afforded by a dense array of millions of nanoscale memory devices serving as compute units. By introducing physical coupling between the memory devices, there is also a potential for further reduction in computational time complexity.[18,19] Memristive devices, such as PCM, ReRAM, and MRAM,[20,21] are particularly well suited for in-memory computing.

There are several key physical attributes that enable in-memory computing using memristive devices. First of all, the ability to store two levels of resistance/conductance values in a non-volatile manner and to reversibly switch from one level to the other (binary storage capability) can be exploited for computing. **Figure 3**a shows the resistance values achieved upon repeated switching of a representative memristive device (a PCM device) between LRS and HRS. Due to the LRS and the HRS, resistance could serve as an additional logic state variable. In conventional CMOS, voltage serves as the single logic state variable. The input signals are processed as voltage signals and are output as voltage signals. By combining CMOS circuitry with memristive devices, it is possible to exploit the additional resistance state variable. For example, the HRS state could indicate logic "0," and the LRS state could denote logic "1." This enables logical operations that rely on the interaction between the voltage and resistance state variables and could enable the seamless integration of processing and storage. This is the essential idea behind memristive logic, which is an active area of research.[24–26] Memristive logic has the potential to impact application areas, such as image

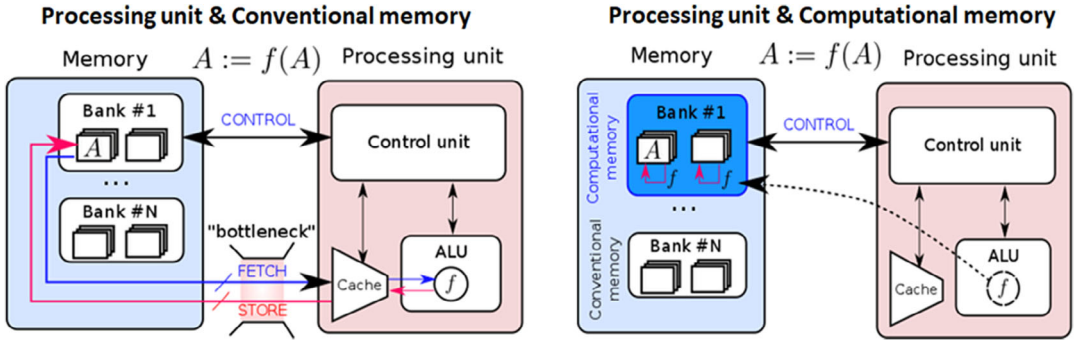


Figure 2. In-memory computing. In a conventional computing system, when an operation f is performed on data D, D has to be moved into a processing unit. This incurs significant latency and energy cost and creates the well-known von-Neumann bottleneck. With in-memory computing, $f(D)$ is performed within a computational memory unit by exploiting the physical attributes of the memory devices. This obviates the need to move D to the processing unit. Adapted with permission.[14] Copyright 2017, Springer Nature.

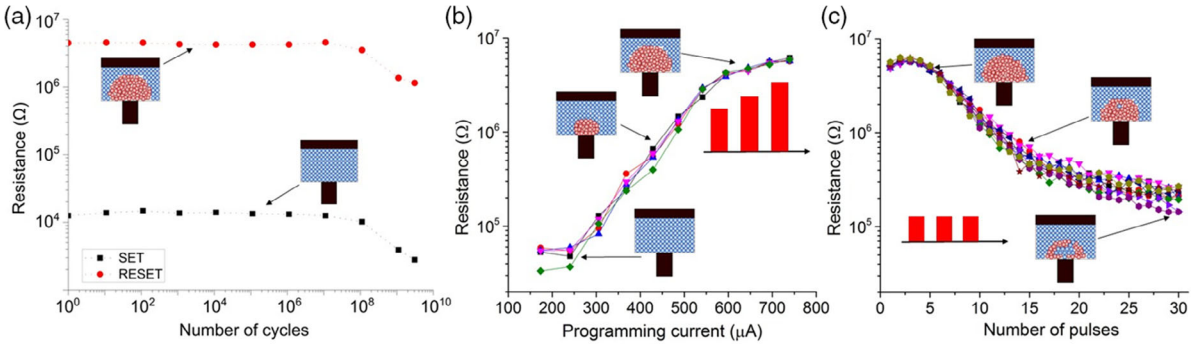


Figure 3. The key physical attributes of memristive devices that facilitate in-memory computing. a) Binary storage capability whereby the devices can be switched between high and low resistance values in a repeatable manner. Adapted with permission.[22] Copyright 2019, IOP Publishing. b) Multi-level storage capability whereby the devices can be programmed to a continuum of resistance values by the application of appropriate programming pulses. Adapted with permission.[23] Copyright 2018, American Institute of Physics. c) The accumulative behavior whereby the resistance of a device can be progressively decreased by the successive application of identical programming pulses. Reproduced with permission.[23] Copyright 2018, American Institute of Physics.

processing,[27] encryption, and database query.[28] Brain-inspired hyper-dimensional computing that involves the manipulation of large binary vectors has recently emerged as another promising application area for in-memory logic.[29,30] Going beyond binary storage, certain memristive devices can also be programmed to a continuum of resistance or conductance values (analog storage capability). For example, Figure 3b shows a continuum of resistance levels in a PCM device achieved by the application of programming pulses with varying amplitude. The device is first programmed to the fully crystalline state, after which RESET pulses are applied with progressively increasing amplitude. The device resistance is measured after the application of each RESET pulse. Due to this property, it is possible to program a memristive device to a certain desired resistance value through iterative programming by applying several pulses in a closed-loop manner.[31] Yet another physical attribute that enables in-memory computing is the accumulative behavior exhibited by certain memristive devices. In these devices, it is possible to progressively reduce the device resistance by the successive application of SET pulses with the same amplitude. Also, in certain cases, it is possible to progressively increase the resistance by the successive application of RESET pulses. Experimental measurement of this accumulative behavior in a PCM device is shown in Figure 3c. This accumulative behavior is central to applications, such as training deep neural networks (DNNs), which is described later. Furthermore, the behavior is not limited to PCM devices, and most memristor-based technologies show potential for multi-level switching and gradual resistance modulation. Therefore, the applications presented using this feature in PCM technology could, in principle, be achieved using most other memristor technologies. We refer readers to dedicated review articles and previous literature that cover gradual resistance modulation obtained in different memristor techniques.[10] The intrinsic stochasticity associated with the switching behavior in memristive devices can also be exploited for in-memory computing.[32] Applications include stochastic computing[33] and physically unclonable functions.[34] We will discuss these concepts in more detail later in the text.

A very useful in-memory computing primitive enabled by the binary and analog non-volatile storage capability is matrix-vector multiplication (MVM).[7,35] The physical laws that are exploited to perform this operation are Ohm's law and Kirchhoff's current summation laws. For example, to perform the operation $Ax = b$, the elements of A are mapped linearly to the conductance values of memristive devices organized in a crossbar configuration. The x values are mapped linearly to the amplitudes of read voltages and are applied to the crossbar along the rows. The result of the computation, b, will be proportional to the resulting current measured along the columns of the array. The concept is shown in **Figure 4** and can be summarized by a simple equation that relates the voltage vector, $\mathbf{V}$, the conductance matrix, $\mathbf{G}$, and the current vector, $\mathbf{I}$, as $\mathbf{I} = \mathbf{G}\ \mathbf{V}$.

Compressed sensing and recovery are one of the applications that could benefit from an in-memory computing unit that performs MVMs. The objective behind compressed sensing is to acquire a large signal at sub-Nyquist sampling rate and to subsequently reconstruct that signal accurately. Unlike most other compression schemes, sampling and compression are done simultaneously, with the signal getting compressed as it is sampled. Such techniques have widespread applications in the domain of medical imaging, security systems, and camera sensors. The compressed measurements can be thought of as a mapping of a signal x of length N to a measurement vector y of length $M < N$. If this process is linear, then it can be modeled by an $M \times N$ measurement matrix M. The idea is to store this measurement matrix in the in-memory computing unit, with memristive devices organized in a crossbar configuration (see **Figure 5**a). In this manner, the compression operation can be performed in $O(1)$ time complexity. To recover the original signal from the compressed measurements, an approximate message passing (AMP) algorithm can be used, using an iterative algorithm that involves several MVMs on the very same measurement matrix and its transpose. In this way, the same matrix that was coded in the in-memory computing unit can also be used for the reconstruction, reducing reconstruction complexity from $O(M \times N)$ to $O(N)$. An experimental illustration of compressed sensing

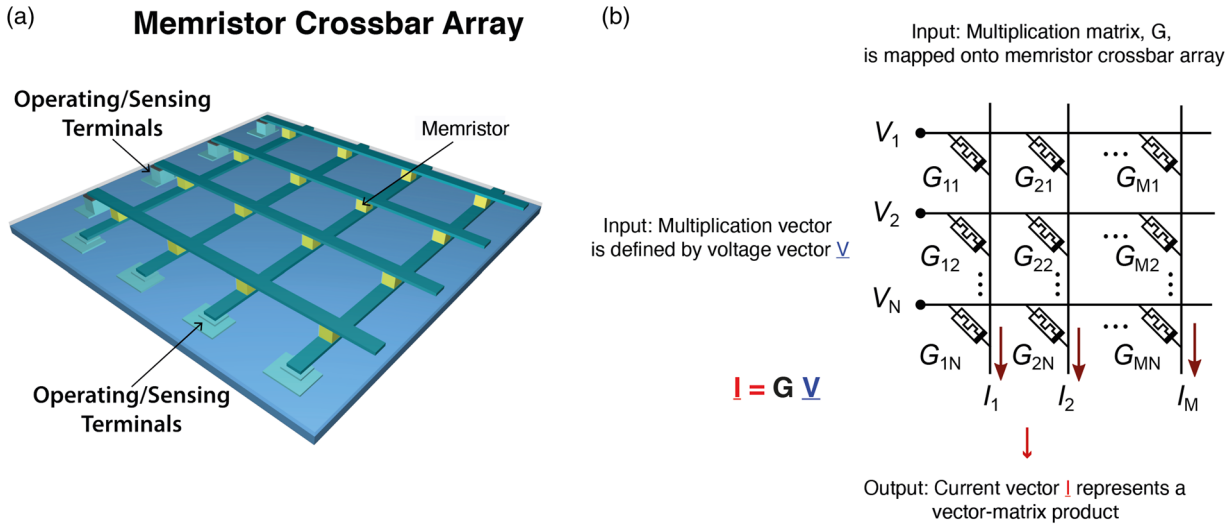


Figure 4. a) Memristor crossbar array. b) Applied (read) voltages and conductance of memristor devices in a crossbar array define an input vector and an input matrix, whereas sensed currents provide a resulting vector of vector-matrix multiplication.

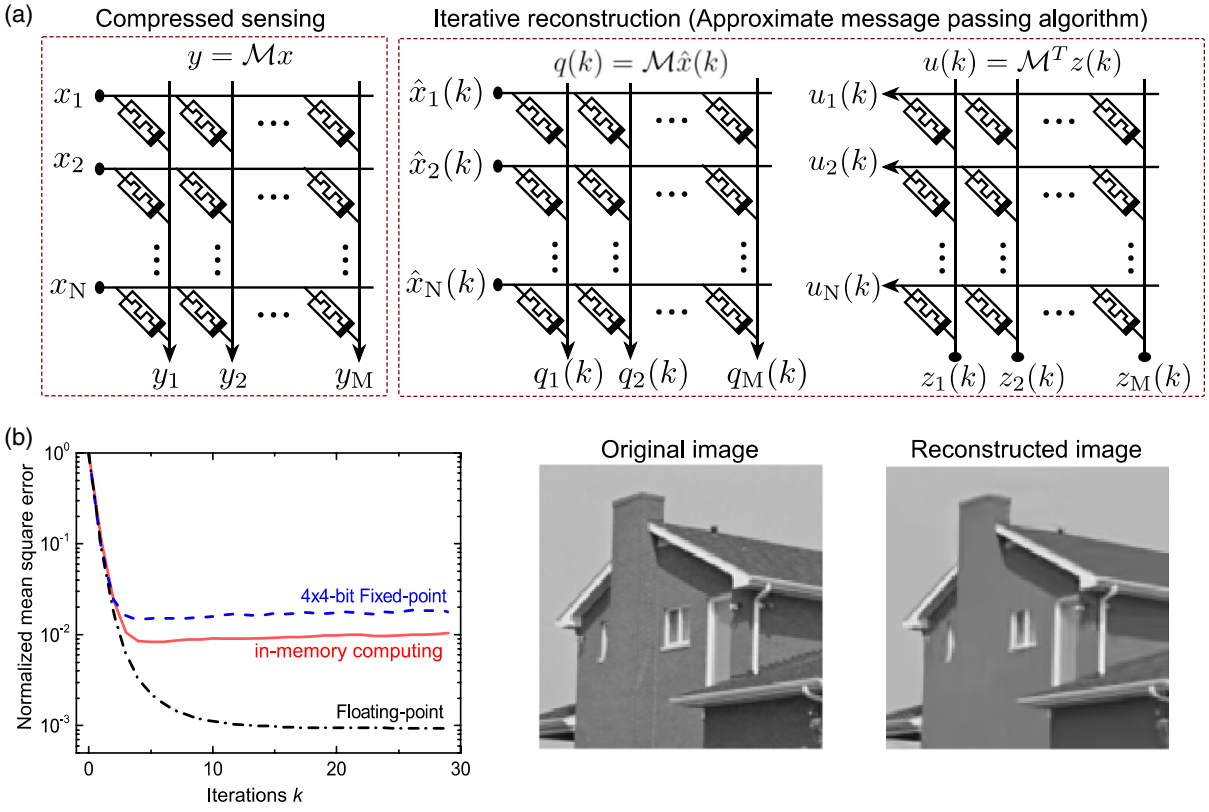


Figure 5. a) Compressed sensing involves one MVM. Data recovery is performed via an iterative scheme, using several MVMs on the very same measurement matrix and its transpose. b) An experimental illustration of compressed sensing recovery in the context of image compression is presented, showing 50% compression of a 128 × 128 pixel image. The NMSE associated with the reconstructed signal is plotted against the number of iterations. Adapted with permission.[36] Copyright 2018, IEEE.

recovery in the context of image compression is shown in Figure 5b. A 128 × 128 pixel image was compressed by 50% and recovered using the measurement matrix elements encoded in a PCM array. The normalized mean square error (NMSE) associated with the recovered signal is plotted as a function of the number of iterations. A remarkable property of AMP is that

its convergence rate is independent of the precision of the MVMs. The lack of precision only results in a higher error floor, which may be considered acceptable for many applications. Note that, in this application, the measurement matrix remains fixed, and hence, the property of PCM that is exploited is the multi-level storage capability.

3. DL Accelerators

DNNs, loosely inspired by biological neural networks, consist of parallel processing units called neurons interconnected by plastic synapses. By tuning the weights of these interconnections using millions of labeled examples, these networks are able to perform certain supervised learning tasks remarkably well. These networks are typically trained via a supervised learning algorithm based on gradient descent. During the training phase, the input data are forward propagated through the neuron layers with the synaptic networks performing multiply accumulate operations. The final layer responses are compared with input data labels, and the errors are backpropagated. Both steps involve sequences of MVMs. Subsequently, the synaptic weights are updated to reduce the error. This optimization approach can take multiple days or weeks to train state-of-the-art networks on conventional computers. Hence, there is a significant effort toward the design of custom ASICs based on reduced precision arithmetic and highly optimized dataflow.[13,37] However, the need to shuttle millions of synaptic weight values between the memory and processing unit remains a key performance bottleneck, both for power and time efficiency, and, hence, in-memory computing is being explored as an alternative approach for both inference and training of DNNs.[38,39] The essential idea is to map the various layers of a neural network to an in-memory computing unit where memristive devices are organized in a crossbar configuration (see **Figure 6**). The synaptic weights are stored in the conductance state of the memristive devices, and the propagation of data through each layer is performed in a single step by inputting the data to the crossbar rows and deciphering the results at the columns.

DL inference refers to just the forward propagation in a DNN once the weights have been learned. Both binary and analog storage capability of memristive devices can be exploited for the MVM operations associated with the inference operation. The key challenges are the inaccuracies associated with programming the devices to a specified synaptic weight as well as drift, noise, etc., associated with the conductance values.[40] Due to this programming noise, the synaptic weights that are obtained by training in high precision arithmetic (e.g., 32 bit floating point) cannot be mapped directly to computational memory. However, it can be shown that by customizing the training procedure to make it aware of these device-level non-idealities, it is possible to obtain synaptic weights that are suitable for being mapped to an in-memory computing unit.[39] For conductance drift, global scaling procedures or periodic calibration of the batch normalization parameters have been found to be very effective.[39] **Figure 7** shows the mixed hardware/software experimental results using a prototype multi-level PCM chip. The synaptic weights are mapped to PCM devices organized in a two-PCM differential configuration (723 444 PCM devices in total). The differential configuration means that two memristors are used per synaptic weight, so both positive and negative weights can be represented. It can be seen that with a hardware-aware custom training scheme, it is possible to approach the floating-point baseline. The temporal decline in accuracy is attributed to the conductance drift exhibited by PCM devices.[41] However, with appropriate compensation schemes, it is possible to maintain software equivalent accuracies over a substantial period of time.

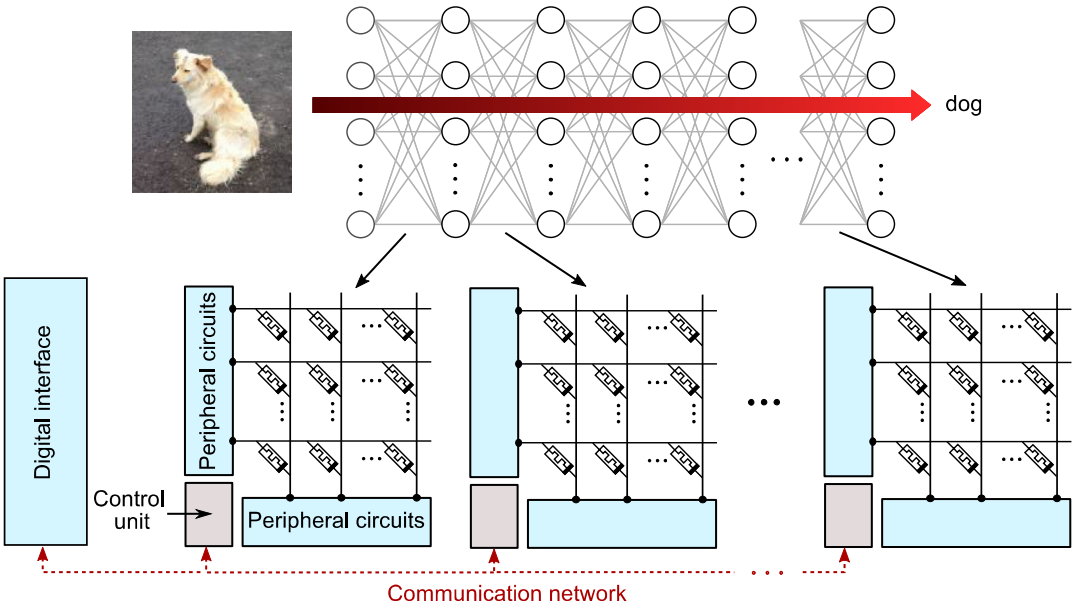


Figure 6. DL based on in-memory computing. The various layers of a neural network are mapped to a computational memory unit where memristive devices are organized in a crossbar configuration. The synaptic weights are stored in the conductance state of the memristive devices. A global communication network is used to send data from one array to another. Adapted with permission.[17] Copyright 2020, Springer Nature.

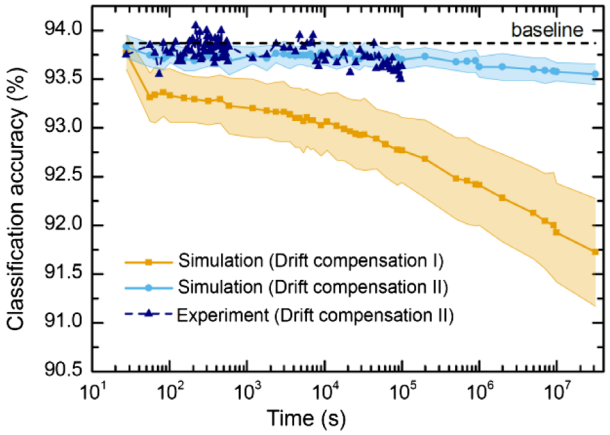


Figure 7. DL inference. Experimental results on ResNet-32 using the CIFAR-10 data set. The classification accuracies obtained via the direct mapping and custom training approaches are compared with the floating-point baseline. Reproduced with permission.[39] Copyright 2020, Springer Nature.

In-memory computing can also be used in the context of supervised training of DNNs with backpropagation. When performing training of a DNN encoded in crossbar arrays, forward propagation is performed in the same way as inference described earlier. Next, backward propagation is performed by inputting the error gradient from the subsequent layer onto the columns of the current layer and deciphering the result from the rows. Subsequently, the error gradient is computed. Finally, the weight update is performed based on the outer product of activations and error gradients of each layer. This weight update relies on the accumulative behavior of memristive devices. Recent DL research shows that when training DNNs, it is possible to perform the forward and backward propagations rather imprecisely, whereas the gradients need to be accumulated in high precision.[42] This observation makes the DL training problem amenable to the mixed-precision in-memory computing approach that was recently proposed.[43] The in-memory compute unit is used to store the synaptic weights and to perform the forward and backward passes, whereas the weight changes are accumulated in high precision (**Figure 8**a).[45,46] When the accumulated weight exceeds a certain threshold, pulses are applied to the corresponding memory devices to alter the synaptic weights. This approach was tested using the handwritten digit classification problem based on the MNIST data set. A two-layered neural network was used with two-PCM devices in differential configuration (≈400 000 devices) representing the synaptic weights. Resulting test accuracy after 20 epochs of training was ≈98% (Figure 8b). After training, inference on this network was performed for over a year with marginal reduction in the test accuracy. The crossbar topology also facilitates the estimation of the gradient and the in-place update of the resulting synaptic weight all in $O(1)$ time complexity.[38,47] By obviating the need to perform gradient accumulation externally, this approach could yield better performance than the mixed-precision approach. However, significant improvements to the memristive technology, in particular, the accumulative behavior, are needed to apply this to a wide range of DNNs.[48,49]

Compared with the charge-based memory devices that are also used for in-memory computing,[50–52] a key advantage of memristive devices is the potential to be scaled to dimensions of a few nanometers.[53–57] Most of the memristive devices are also suitable for back end of line integration, thus enabling their integration with a wide range of front-end CMOS technologies. Another key advantage is the non-volatility of these devices that would obviate the need for computing systems to be constantly connected to a power supply. However, there are also challenges that need to be overcome. The significant intra-device and intra-device variability associated with the LRS and HRS states is a key challenge for applications where memristive devices are used for logical operations. For applications that rely on analog storage capability, a significant challenge is programming variability that captures the inaccuracies associated with programming an array of devices to desired conductance values. In ReRAM, this variability is attributed mostly to the stochastic nature of filamentary switching, and one prominent approach to counter this is that of establishing preferential paths for conductive filament formation.[58,59] Representing single computational elements using multiple memory devices is another promising approach.[60] Yet another challenge is the temporal and temperature-induced variations of the programmed conductance values. The resistance "drift" in PCM devices, which is attributed to the intrinsic structural relaxation of the amorphous phase, is an example. The concept of

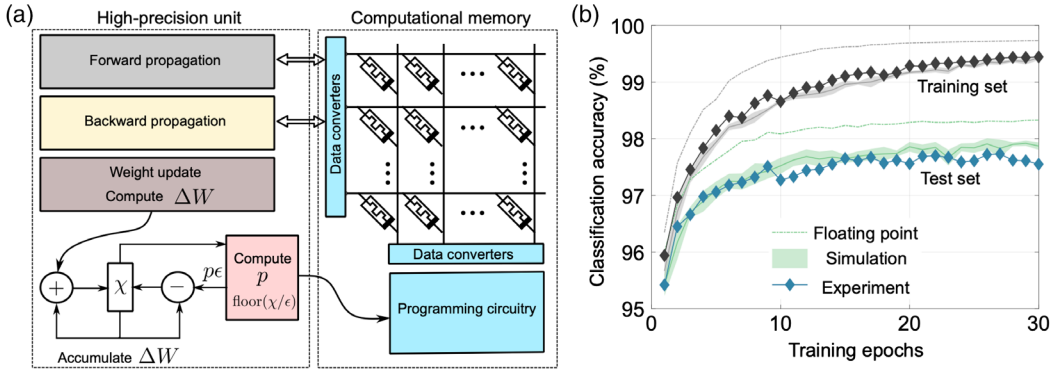


Figure 8. DL training. a) Schematic illustration of the mixed-precision architecture for training DNNs. b) The synaptic weight distributions and classification accuracies are compared between the experiments and floating point baseline. Reproduced with permission.[44] Copyright 2020, Frontiers Media.

projected PCM is a promising approach toward tackling "drift."[61,62] Other memristor technologies could have similar issues related to states retention or occurrence of random telegraphic noise. The requirements that the memristive devices need to fulfill when used for computational memory are heavily application-dependent. For memristive logic, high cycling endurance ($>10^{12}$ cycles) and low device-to-device variability of the LRS/HRS resistance values are critical. For computational tasks involving read-only operations, such as MVM, it is required that the conductance states remain relatively unchanged during their execution. It is also desirable to have a gradual analog-type switching characteristic for programming a continuum of resistance values in a single device. A linear and symmetric accumulative behavior is also required in applications where the device conductance needs to be incrementally updated such as in DL training.[63] For stochastic computing applications, random device variability is not problematic, but any device degradation should be gradual and graceful to compensate for variations in switching voltages.[64] We further summarize some common device non-idealities and proposed approaches to minimize the adverse effects in Section 6.

4. SNNs and Memristors

As opposed to the DL networks discussed earlier, SNNs can more naturally incorporate the notion of time in signal encoding and processing. SNNs are typically modeled on the integrate-and-fire behavior of neurons in the brain. In this framework, neurons communicate with each other using binary signals or spikes. The arrival of a spike at a synapse triggers a current flow into the downstream neuron, with the magnitude of the current weighted by the effective conductance of the synapse. The incoming currents are integrated by the neuron to determine its membrane potential, and a spike is issued when the potential exceeds a threshold. This spiking behavior can be triggered in a deterministic or probabilistic manner. Once a spike is issued, the membrane potential is reset to a resting potential or decreased according to some predetermined rule. The integration is limited to a specific time window, or else a leak factor is incorporated in the integration, endowing the neuron model with a finite memory of past spiking events.

Compared with the realization of the second-generation DNNs (discussed in the previous section), SNNs can potentially have significant improvements in efficiency. The first reason for this comes from the underlying signal encoding mechanism. The calculation of the output of a neuron involves the determination of the weighted sum of synaptic weights with real-valued neuronal outputs of the previous layer. For a fully connected second-generation DNN with N neurons in each layer, this requires N^2 multiplications of real-valued numbers, typically stored in low precision representations. In contrast, the forward propagation operation in an SNN only requires addition operations, as the input neuronal signals are binary spike signals. To elaborate, assume that the input signal is encoded as a spike train with duration T, with a minimum inter-spike interval of Δt. If the probability of a spike at any instant of time is p, then, on an average, $NpT/\Delta t$ spikes have to be propagated through the synapses, and this requires $N^2pT/\Delta t$ addition operations. In most modern processors, the cost of multiplication, C_m, is 3–4 times higher than that of addition, C_a. Hence, provided the neuronal and synaptic variables required for computation are available in the processor, SNNs offer a path to more efficient computation if the inequality

$$C_a p\left(\frac{T}{\Delta t}\right) < C_m \tag{1}$$

holds. Hence, it is important to develop algorithms for SNNs that minimize p and $(T/\Delta t)$ to improve computational efficiency. This requires the use of sparse binary signal encoding schemes that go beyond rate coding that is typically used in SNNs today. The following section will discuss strategies to develop general-purpose learning rules for SNNs that satisfy such constraints. The second potential for efficiency improvement of SNNs arises because of novel memristor-based processor architectures. While SNNs can be implemented using Si CMOS static random-access memory or dynamic random-access memory technologies, the advent of novel nanoscale memristive devices provides opportunities for significant improvements in overall computational efficiency.

As mentioned in the previous section, memristive devices can be integrated at the junctions of crossbar arrays to represent the weights of synapses, and CMOS circuits at the periphery can be designed to implement the neuronal integration and learning logic. The small form factor of the devices, coupled with the scalability of operating voltages and currents beyond what is possible with conventional CMOS, suggests that these architectures can have several orders of magnitude efficiency improvement over silicon-based implementations.[65,66] However, apart from the already mentioned non-idealities of memristive devices, crossbar arrays with more than 2048×2048 devices present reliability issue due to the resistance drop on the wires and the sneak paths that corrupt the measurement and programming of synaptic states. One approach to mitigate these issues is to design neurosynaptic cores with smaller crossbars and associated neuron circuits, tile these cores on a 2D array, and provide communication fabrics between the cores.[67] Such tiled neurosynaptic core-based designs are particularly amenable for realizing SNNs, as only binary spikes corresponding to intermittently active spiking neurons need to be transported between cores, as opposed to real-valued neuronal variables that are active for all the neurons in the core in the case of DL networks. This is the second inherent advantage that SNNs have over DNNs for computational efficiency improvement.

Overcoming the reliability challenges mentioned earlier is essential for building reliable systems and would require the co-optimization of algorithms and architectures that are designed to mitigate or leverage these non-ideal behaviors for computation. Two kinds of systems can be visualized based on the application mode. Inference engines, which do not support on-chip learning, can be designed based on memristive devices integrated on crossbars, where the devices are programmed to the desired conductance states based on the weights obtained from software training. However, as memristive devices support incremental conductance changes by the application of suitable electrical programming pulses, it is also possible to design learning systems where network weight updates are implemented on-chip in an event-driven manner.[68] There are also many recent

examples where these devices have been engineered to mimic the integration and fire characteristics of biological neurons,[69–71] potentially enabling the realization of all-memristor implementations of SNNs.[72] The field is still in its infancy and, so far, has only witnessed small proof-of-concept demonstrations. We now discuss some of the approaches that have been explored toward realizing memristive-based inference-only spiking networks as well as learning networks with SNNs.

4.1. Memristive SNNs for Inference

A common approach to develop SNNs is to start with a second-generation ANN trained using traditional backpropagation-based methods, and then convert the resulting network to a spiking network in software. These solutions are based on weight-normalization schemes, so that the spike rates of the neurons in the SNN are proportional to the activations of the neurons in the ANN.[73,74] While this should, in principle, result in SNNs with comparable accuracies as their second-generation counterparts, some device-aware re-training would typically be necessary when the network is implemented in hardware due to the non-linearity and limited dynamic ranges of nanoscale devices.

One of the differentiating features of inference engines is that the nanoscale devices storing state variables are programmed only rarely, compared with the number of reads (potentially at every inference cycle). As higher energy programming cycles have a stronger effect in degrading device lifetimes compared with the lower energy read cycles, this mode of operation can have better overall system reliability compared with that of learning systems.

In a preliminary hardware demonstration leveraging this approach, Midya et al. used memristors based on SiO_xN_y:Ag to implement compact oscillatory neurons whose output voltage oscillation frequency is proportional to the input current.[75] In this proof-of-concept demonstration of a three-layer network, ANN to SNN conversion was limited to the last layer alone, but the approach could be extended to hidden layers as well.

4.2. Memristive SNNs for Unsupervised Learning and Adaptation

Most hardware demonstrations of SNNs using memristive devices have focused on the unsupervised learning paradigm, where the synaptic weights are modified in an unsupervised manner according to the biologically inspired STDP rule.[76] The rule captures the experimental observation that when a synapse experiences multiple pre-before-post pairings, the effective synaptic strength increases, and conversely, multiple post-before-pre spike pairs result in an effective decrease in synaptic conductance.

It should be noted that while other biological mechanisms may also play a key role in learning and memory formation in the brain, as have been observed experimentally,[77,78] STDP is a simple local learning rule, which is especially straightforward to implement in hardware. While it is possible to implement-timing-dependent plasticity rules using many-transistor CMOS circuits,[79] it was experimentally demonstrated early on that memristive devices can exhibit STDP-like weight adaptation behaviors upon the application of suitable waveforms.[68,80,81] Going beyond individual device demonstrations, IBM has also demonstrated an integrated neuromorphic core with 256×256 PCM synapses fabricated along with Si CMOS neuron circuits capable of on-chip learning based on a simplified model of STDP for auto-associative pattern learning tasks.[82]

Boybat et al. used phase change memristive synapses to demonstrate temporal correlation detection through unsupervised learning based on a simplified form of STDP,[60] as shown in **Figure 9**. In their experiment, a multi-memristive architecture was introduced, where N PCM devices are used to represent one synapse, with all devices within a synapse read during spike transmission, but only one of the devices, selected through an arbitration scheme, is programmed to update the synaptic weight. Software equivalent accuracies could be obtained in the experiment with this scheme, although the individual devices are plagued by several common non-ideal effects, such as programming non-linearity, read noise, and conductance drift. Note that with $N = 1$ device representing a synapse, the network

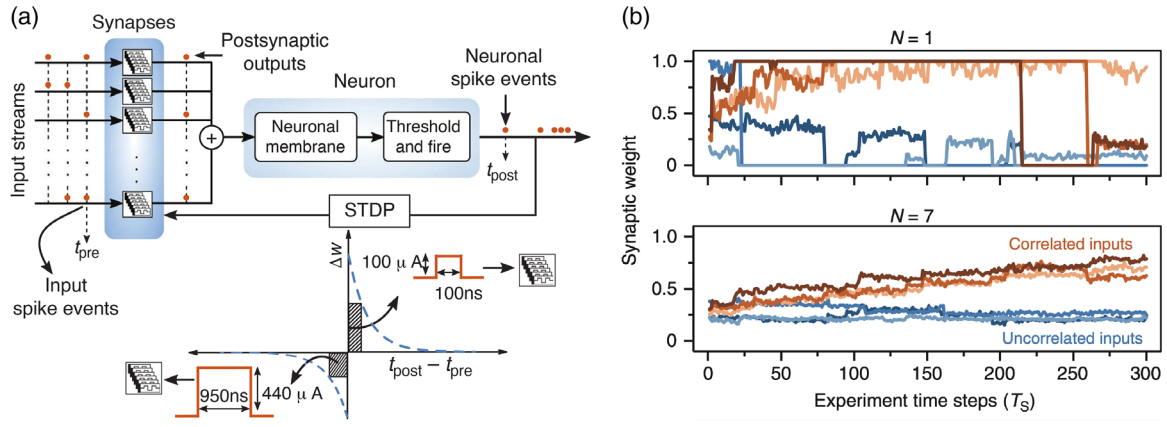


Figure 9. a) Unsupervised learning demonstration using multi-memristive PCM architecture. The network consists of an integrate and fire neuron receiving inputs from 1000 multi-PCM synapses, with each synapse being excited by Poisson generated binary spike streams. 10% of the synapses receive correlated inputs, whereas the rest receive uncorrelated inputs. The weights evolve based on the simplified STDP rule shown. b) With $N = 7$ PCM device per synapse, the correlated and uncorrelated synaptic weights evolve to well-separated values, whereas with $N = 1$, the separation is corrupted due to programming noise. Reproduced with permission.[60] Copyright 2018, Springer Nature.

accuracy was significantly lower than the software baseline; $N = 7$ devices were necessary to obtain close to ideal performance.

Spiking networks can also be used for other unsupervised learning[83] and adaptation tasks. Recently, Fang et al. demonstrated that certain optimization problems could be solved driven by the coupled dynamics of ferroelectric field-effect transistor (FeFET)-based spiking neurons.[84] While there was no synaptic weight adaptation in this approach, the optimal solution to the problem is determined by the coupled interactions between the neurons, which modulate each other's membrane potentials in an event-driven manner.

4.3. Memristive SNNs for Supervised Learning

Compared with the previous two approaches, implementing supervised learning in SNNs is a more challenging task, as the algorithm and the network must generate spikes at precise time instants based on the input excitation. As opposed to the backpropagation algorithm that is highly successful in training ANNs, supervised learning algorithms for SNNs are not well developed yet, due to the inherent difficulty in applying gradient descent methods for spiking neuron models with infinite discontinuities at the instants of spikes. Nevertheless, there have been several demonstrations of supervised learning algorithms for SNNs based on approximate forms of gradient descent for simple fully connected networks.[85–87]

Recently, Nandakumar et al. demonstrated a proof-of-concept realization of supervised learning in a two-layer SNN implemented using nanoscale PCM synapses based on the Normalized Approximate Descent Algorithm.[88] In the experiment, 132 spike streams representing spoken audio signals generated using a silicon cochlea chip were used as the input, and the network was trained to generate 168 spike streams whose arrival times indicate the pixel intensity corresponding to the spoken characters.[88] Compared with normal classification problems in deep networks where the accuracy depends only on the relative magnitude of the response of the output neurons, the SNN problem is harder, as the network is tasked with generating close to 1000 spikes at specific time instances over a period of 1250 ms from 168 spiking neurons that are excited by 132 input spike streams. The accuracy for spike placement obtained in the experiment was about 80% compared with the software baseline accuracy of over 98%, despite using the same multi-memristive architecture described earlier. This experiment is, hence, illustrative of the need for developing more robust and event-driven learning algorithms for SNNs that can mitigate or even leverage the device non-idealities for designing computational systems (**Figure 10**).

In summary, spike-based learning and inference are promising facets of the neuromorphic computing paradigm. Unlike conventional ML models, spike-based processing "computes with time, not in time."

5. Some Challenges of Memristor Technologies

While memristor technology shows a huge promise for a wide range of applications, several significant challenges need to be addressed to make them commercially viable. It is essential to recognize that different applications have different device requirements. There are general challenges related to reliability, fabrication, uniformity, and scalability that need to be addressed regardless of the application. The family of memristor technologies is diverse, and different technologies come with different kinds of device and system non-idealities. These are covered extensively in the available literature,[89] where scalability and reliability issues are discussed in the context of specific memristor technologies. While some of the stringent requirements typically necessary for non-volatile data storage and memory applications might be relaxed for analog and neuromorphic computing, other additional device properties might be specifically required. Here, we will discuss some key examples of typical non-idealities relevant for the types of computing applications covered in this report, as well as some approaches that may be used to mitigate them.

One of the most prominent issues relates to the non-linearity in current/voltage characteristic seen many memristive devices. This prevents accurate vector-matrix computation using memristive crossbars, as the output current does not depend linearly on the applied voltage, and the linear relationship ($\mathbf{I} = \mathbf{G}\,\mathbf{V}$) assumed cannot be used over the whole voltage range. Few techniques

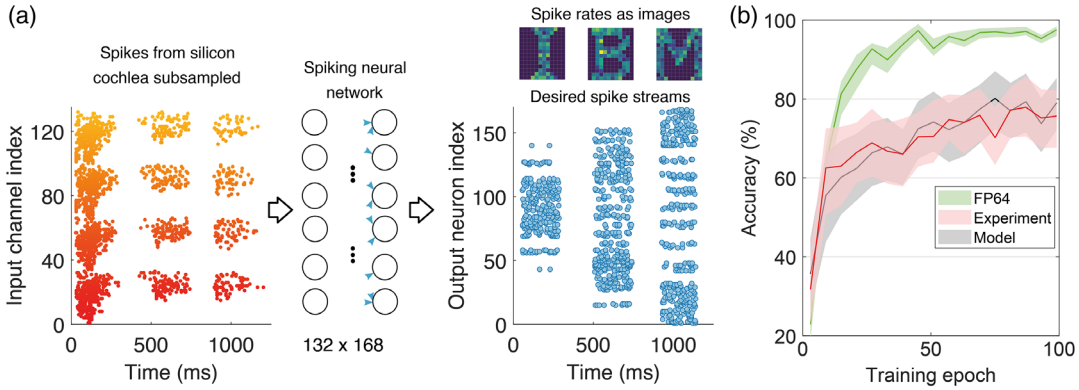


Figure 10. a) SNN supervised learning experiment. A two-layer network is tasked with generating 1000 ms long spike streams from the 168 neurons at the output corresponding to the images of the spoken characters. The inputs to the network are 132 spike streams representing the characters subsampled from the output of a silicon cochlea chip. The weights are modified based on the NormAD learning rule. b) Using multi-PCM synapses, the accuracy of spike placement at the output is about 80%, compared with the FP64 accuracy of close to 98%. Reproduced with permission.[88] Copyright 2020, Springer Nature.

have been proposed to deal with I/V non-linearities such as hot-forming step prior to programming,[90] or use of transistor selector elements (1T1R architectures).[91] By elevating the temperature to 150 °C during the electroforming step, it is possible to increase the number of oxygen vacancies generated in Ta_2O_5 ReRAM devices and form denser conductive filaments. Denser filaments formed at the elevated temperatures exhibit much better I/V linearity compared with those formed at the room temperature. Using a transistor as a selector element in the 1T1R architecture allows for the much better control of the current compliance during the set and the electroforming processes. This is a likely reason for the improved I/V linearity.

The next non-ideality concerns the cycle-to-cycle and device-to-device variability that is inherent to nanoscale devices. This could be addressed partially by improved materials engineering, such as the use of ultra-thin atomic layer deposition-TiN buffer layers,[92] or the engineering of oxide–metal interfaces[93,94] and oxide microstructure[95] that improves the stability of operating characteristics. These techniques are used to restrict filaments formation to the particular sites and limit significant variations in filaments configuration from cycle to cycle. Faulty devices could be considered by adapting mapping schemes (the way that weights in ANNs are mapped onto the memristor conductance states), and using redundancy techniques.[96] The issue of limited dynamic range (i.e., separation between on- and off-state conductance) could be extended using two or more devices whose conductances are assigned different significances to represent a single weight.[49] Adverse effects that arise due to the finite resistance of metal wires used for the crossbar are mitigated using advanced mapping or compensation schemes.[97]

Recently, a technology agnostic technique called committee machines (CMs) has demonstrated significant potential in increasing the inference accuracy when dealing with several non-idealities.[98] The approach does not assume any prior knowledge of particular non-idealities or the use of customized training procedures. The main idea behind CMs is to use committees of smaller neural networks and to average the inference outputs. This leads to higher inference accuracy, even with the same total number of devices used (the total number of weights from all members of the committee is equal to the number of weights in a single large neural network). This is shown in **Figure 11**. Put simply, it is advantageous to combine several smaller neural networks rather than to use a single large neural network when dealing with non-perfect memristor devices.

While the abovementioned requirements are sufficient for building reliable inference engines, on-chip training in DL accelerators also requires that device conductance can be programmed in an a linear fashion. Ideally, it would be possible to increase or decrease the device conductance linearly based on the number of identical (positive or negative) voltage pulses, and not depending on the current state. If the devices could be linearly tuned, simple programming schemes to implement on-chip learning can be used without requiring complex peripheral circuits. However, most memristor technologies exhibit highly non-linear programming characteristics. The resistance change is not only dependent on pulse width/amplitude, but also on the current resistance state. Much work has been done to obtain devices that exhibit linear programmability, as well as to develop specific programming schemes that would improve non-linear resistance modulation. One such scheme is the state-dependent (SD) programming that instead of identical pulses uses exponentially increasing pulse width while keeping the amplitude fixed.[99] Excellent programming linearity can be obtained using this scheme; however, the drawback is that the complexity of peripheral control circuitry increases, limiting the overall power-efficiency gains, as additional memory is required to record current conductance states and to verify them before every programming pulse. Another approach is to use identical pulses consisting of one programming pulse and one offset pulse of opposite polarity. This approach, bipolar scheme (BP), does not require tracking of current resistance states as in the case of SD, and the circuitry is not as complex and limiting.[100] Materials optimization includes the use of bilayers that are shown to improve non-linear programming.[101]

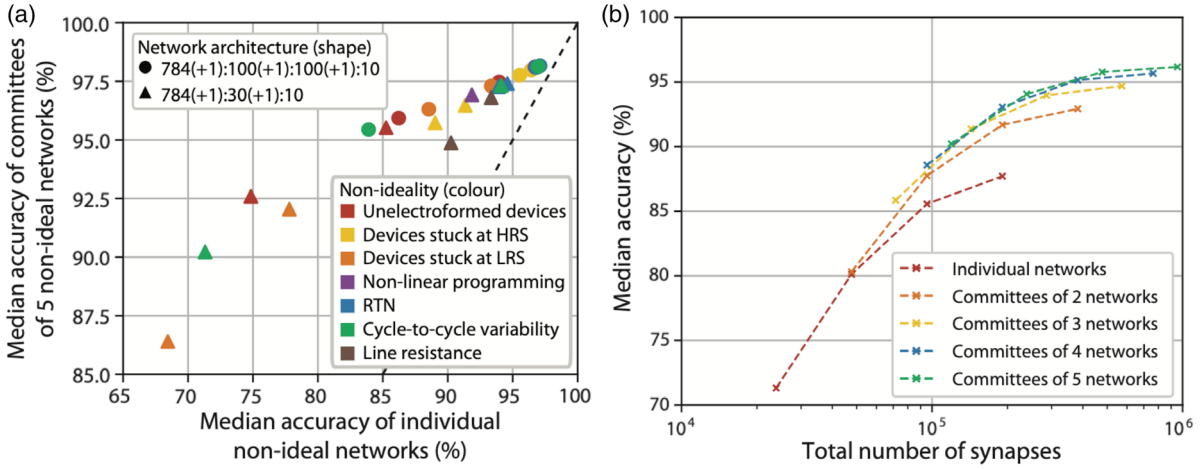


Figure 11. Effectiveness of CMs to dealing with different types of device and system non-idealities. a) Inference accuracy of committees (of size 5) as a function of the accuracy of the individual networks that constitute the committees. All data points above the dashed line indicate improvement in accuracy. b) Median accuracy achieved by individual networks and committees of networks plotted against the total number of synapses required for these neural networks or their committees. Reproduced with permission.[98] Copyright 2019, The Author(s). Published in arXiv.

It is important to note that the majority of non-idealities are studied in single devices or small arrays, and the available statistics is still limited. For memristor technology to gain full maturity, there is a great need for more extensive reliability studies, where adverse effects from lower tail bits might become more apparent.

6. Harnessing Hardware Randomness for Learning

As discussed in the previous sections, the implementation of standard deterministic systems may be severely impaired in hardware implementations whose components are inherently noisy. This is the case for memristive implementation of deterministic ANNs and SNNs in which the synaptic weights are defined by the conductance of memristors. As also seen, state-of-the-art solutions are predicated on the need to mitigate the adverse effects. In this section, we explore the idea that, if properly harnessed, native hardware randomness can be an asset, rather than a nuisance, for the purpose of developing learning and inference machines.

The main argument in favor of harnessing, rather than mitigating, randomness is that it enables the implementation of the primitive of sampling without the need for specialized hardware. Sampling, that is, drawing random numbers from a probability distribution, is a key step for the deployment of probabilistic models and of Bayesian learning and inference strategies.[102,103] As we will discuss, probabilistic spiking neuron models have potential advantages over conventional deterministic counterparts, such as leaky integrate-and-fire, in facilitating the development of flexible learning rules. Furthermore, as will also see, unlike standard learning strategies, Bayesian learning can quantify uncertainty, enhance generalization, and provide tools for a principled exploration of the parameter space.

There are generally two way to inject noise—at the level of the activation of each neuron and at the level of the synaptic weights. The first approach enables the implementation of probabilistic spiking behavior. For example, the stochasticity of the switching process in ReRAM-based memristor devices has been utilized to this end.[104] Synaptic sampling, instead, allows the deployment of Bayesian learning and inference, and a number of different hardware platforms, including memristors, have been proposed for this purpose.[105–108]

In the following, we first review the role of probabilistic spiking models for learning, and then provide a short discussion of Bayesian methods. This discussion is aimed at offering some guideline on the development of suitable hardware platforms and on the exploration of properties of memristive devices that typically seen as disadvantageous. Although these algorithmic concepts are currently less well explored in direct hardware implementations than standard deterministic methods, we believe that memristor technologies could be particularly well suited for their efficient implementations.

6.1. Probabilistic SNNs

As we have discussed, deterministic spiking neuron models such as leaky integrate-and-fire define non-differentiable functions of the synaptic weights: Increasing or decreasing the synaptic weights of a spiking neuron may cause the membrane potential to cross or step back from the spiking threshold, causing an abrupt change in the output. The derivative with respect to the weights is, hence, zero except around the firing threshold, where it is undefined. As a result, standard gradient-based learning rules cannot be directly derived for deterministic models of SNNs. In probabilistic SNN models, a neuron spikes probabilistically with a probability that increases with its membrane potential. Probabilistic models for SNNs solve the outlined problem by defining as learning criterion a function of the probability of spiking according to the desired spatio-temporal patterns. This function is differentiable in the weight vectors.

To elaborate, in supervised and unsupervised learning, the learning problem can be formulated as the minimization of a loss function that measures the degree to which the spiking behavior of neurons in a readout layer conforms to the desired behavior dictated by the training set.[109,110] This problem is differentiable when defined in terms of the probability of the spiking signals in the readout layer.[105] In reinforcement learning, the goal is to minimize a time-averaged reward signal obtained by the learning agent, as it interacts with the environment, making observations and taking actions. A reinforcement learning agent is, hence, faced with the problem of balancing the need for exploration of the parameter space with that of exploiting its current knowledge to increase the reward signal. This can be done by optimizing over a probabilistic policy that chooses actions with a confidence that increases as training proceeds. The resulting optimization problem can be formulated in terms of the probability of the behavior of the neurons in the readout layer when the latter is converted into actions.[110]

Once a learning criterion is determined based on the problem under study, because of the differentiability of the function to be optimized, training can be carried out via stochastic gradient-based rules.

As a related note, we observe that another advantage of probabilistic SNN models is that they can be directly extended with minor conceptual and algorithmic difficulties to allow for multi-valued spikes or inter-neuron instantaneous connections or, equivalently, Winner Take All (WTA) circuits.[111] This is particularly important, because data produced by some neuromorphic sensors incorporate a sign to indicate a positive or negative change.[112] Furthermore, various decoding rules, such as first-to-spike, can be directly optimized for, instead of having to rely on surrogate target spiking sequences.[113]

To illustrate the potential advantages of probabilistic SNN models, we consider a standard reinforcement learning task, in which a learning agent acts in a grid world with the aim of finding the shortest way to an unknown goal location. Two approaches are compared. The first is the standard method of training an ANN model and converting the trained weights for use in an SNN with the same architecture. The alternative approach directly trains a probabilistic SNN as a stochastic policy that selects actions, i.e., moves in the grid world, by trading exploration and exploitation. **Figure 12** compares the performance of these two solutions as a function of the resolution of the input grid representation. The results clearly validate the intuition that directly training the stochastic policy, as a probabilistic SNN is

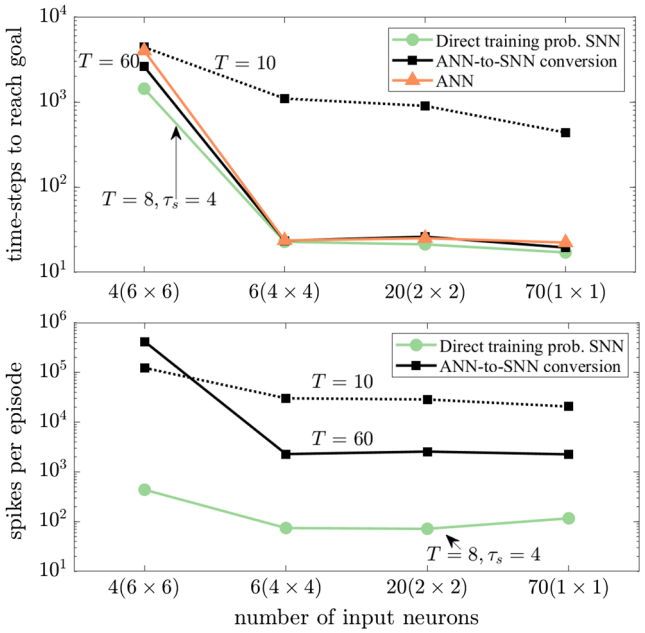


Figure 12. Consider the standard reinforcement learning task in which a learning agent aims at finding a shortest path to an unknown goal point in a grid world through episodic interactions with the environment. The figure shows the time steps needed to reach the goal and the number of spikes per episode for the standard approach of converting a pre-trained ANN and for the direct training or a probabilistic SNN. Reproduced with permission.[110] Copyright 2019, IEEE.

more effective, as well as efficient in terms of number of spikes, than using ANN-to-SNN conversion.

6.2. Bayesian SNNs

As mentioned, synaptic sampling enables the implementation of Bayesian inference and learning.[105–108] Unlike the conventional approach considered thus far of identifying a single set of parameter vectors during learning, the Bayesian principle prescribes the inference of a probability distribution over the synaptic weights. While in the presence of sufficient data, this distribution concentrates on the optimal weight configuration, when data are limited, the synaptic weight distribution provides a "credibility" profile in the parameter space. This, in turn, allows the assessment of uncertainty and of the confidence of the model's decisions or actions, as well as the principled exploration of the parameter space.

Efficient Bayesian learning methods rely on the capacity of the model to draw samples from the current weight distribution. This is important both during inference, to obtain a credibility profile over outputs, and during learning, to enable exploration. Initial efforts toward the implementation of Bayesian methods on hardware include references.[105–108] These papers implement synaptic sampling by including additional circuitry. In contrast, we envision that the inherent randomness of switching processes in memristive devices could provide a source of randomness "for free." Memristors may, hence, be the missing piece that will unlock the potential of spike-based computing.

7. Future of Neuromorphic and Bio-Inspired Computing Systems

Taking a "big picture" view, current AI and ML methods, in particular, have achieved astonishing results in every field they have been applied to and have become or are becoming standard tools for nearly every type of industry one can think of. This impressive invasion was mainly propelled by DL, which is loosely inspired by biological neural networks.

DL primarily refers to learning with ANNs of many layers and, fundamentally, is not different to what we know about that field in the 1990s. Indeed, the key algorithm underlining the success of DL, backpropagation, is an old story: "Learning with backpropagating errors" by Rumelhart et al. was published in 1986.[114] The most commonly used neural networks are feedforward neural networks, and convolutional neural networks used for image processing can be seen as inspired by our visual system, and both of these are not very new concepts.

Backpropagation is, perhaps, the most fundamental method we can think of for parameter optimization. It is derived by differentiating an error function with respect to the learnable parameters, so in some ways, it is not entirely surprising that the algorithm existed for many years. What might be somehow surprising is that we have not been able to move away much from this idea. While there has been recent progress, much of it consisted of relatively small additions and tweaks, for instance, new ways to address the so-called "problem of the vanishing gradient," the deterioration of the error signal as is backpropagated from the output to the input of the network. Undoubtedly, there were some fundamentally different architectures, smart techniques, and novel analyses, but, arguably, the key factor behind such a success seems to be the vast availability of data and computational power.

In fact, recent advances of the neuroscience community are not present in the neural networks. We do not want to argue that this, per se, is either good or bad, or to suggest that the next super-algorithms will be copying nature. We only want to underline that though inspired only, ANNs had their basis on neuroscience concepts and that there are many phenomena that have, perhaps, not been sufficiently explored within an AI context. For instance, biological neural networks have different learning rules for positive and negative connections; connections change in multiple time scales and show reversible dynamic behavior (known as short-term plasticity), and the brain itself has a structure where specific areas play different roles, just to name a few.

Instead, our progress was mainly based on hardware improvements that made this success possible by allowing long training phases; an amount of training unrealistic for any human. While it is true that human intelligence also develops over years and that human learning involves many trials, for comparison, AlphaGoZero, which surpass human performance in the game of Go, was trained over 4.9 million games.[115] To match this number of games would require a human that lives for 90 years to complete one Go game every 10 min from the moment they are born. This realization tells us two things: 1) our machines do not learn the same way that humans do, and even if we think our methods as bio-inspired, we likely still miss some key

ingredients and 2) executing that many games certainly requires considerable computational power and energy consumption.

As a consequence, training algorithms often require a high energy footprint due to excessive training times and hyper parameter tuning involved. Hyper-parameters are parameters of the system that are not (usually) adapted via the learning method itself; one such example is the learning rate, which indicate how fast the network should update its "knowledge." Before rushing to say that a high learning rate is obviously desirable, such a learning rate could lead to oscillations as, for instance, optimal solutions could be overlooked, or it could lead to forgetting previously obtained knowledge. Setting the learning rate right is not always trivial. In fact, the tuning of hyper-parameters was what originally made the ML community to turn away from ANNs, and it was the performance of DL that brought the focus back. One may then wonder, at the end of the day how much energy inefficient could DL systems be? The reply is, perhaps, surprising: estimated carbon emissions for training standard natural language processing models is approximately five times higher than running a car for a lifetime.[116] This realization suggests there is an urgent need to improve on both current hardware and learning models.

Given such energy concerns, systems based on low-power memristive devices are a highly promising alternative.[117,118] Besides having a low carbon footprint, many studies demonstrated devices that mimic neurons, synapses, and plasticity phenomena. Often, such approaches work well for off-line training. However, some of these attempts, particularly where plasticity is involved, are opportunistic (including own work), and how scaling to larger networks could happen is not always obvious. Faithfully reproducing the brain functionality, when neuroscience has already so many open questions, is challenging for any technology. Moreover, using technologies that potentially allow less possibilities for engineering in comparison with traditional methods (such as CMOS) might well be mission impossible. How far can we go by reconstructing neuron by neuron and synapse by synapse in terms of scalability remains unclear. A more promising way might be to achieve a deeper understanding of the physics of the relevant materials and based on this understanding co-develop the technology and the required learning methods for achieving AI.

In the meantime, in parallel, we can immediately explore simple bio-inspired approaches that harness the dynamics of the material and could be proven useful for particular sets of problems. Here, we present one such example, which stems from the area of reservoir computing, an idea invented separately by Herbert Jeager for the ML community,[119] under the name of echo state networks, and by Wolfgang Mass[120] for the computational neuroscience community, under the name of liquid state machines. We strongly suspect that both these methods were very much motivated by the difficulty of training recurrent networks with a generalization of backpropagation known as backpropagation through time. While feedforward networks can perform many tasks successfully, recurrences are required for memory, and moreover, the brain is clearly not only feedforward. If recurrences exist and are required, there must be a way to efficiently train such structures. As a side note, it is very difficult to imagine how a biological neural network would be able to implement backpropagation through time, and for this alternative, approaches have recently made their appearance.[121]

Reservoir computing methods came up with a workaround to the problem of training recurrent networks: they do not train them but instead harness their properties. Common in the approaches of echo state networks and liquid state machines is the idea of using a randomly recurrent network with fixed connectivity, hence no need to resort to backpropagation through time. This recurrent network is called a reservoir. It provides memory and at the same time transforms the input data to a spatiotemporal representation of higher dimensionality. This enhanced representation can be used as an input to single-layer perceptrons that are trained with a very simple learning method, so the only learnable parameters are the feedforward weights between the reservoir neurons and the output neurons. The key difference between the echo state networks and liquid state machines is that the first approach uses recurrent artificial neuron dynamics, whereas the second uses recurrent SNNs, reflecting the mindset in their corresponding communities. The main principle of reservoir computing is shown in **Figure 13**. The input $x(t)$ is projected into the higher dimensional feature space $r(t)$ using the dynamical reservoir system. Only the weights connecting the internal states $r(t)$ with the output $y(t)$ need to be trained, whereas the rest of the system is fixed. The advantage of this approach is that it only requires a simple training method, whereas the ability to process complex and temporal data is retained.

Indeed, it might be surprising how much randomness can do from the point of computation: a random network can enrich data representations sufficiently, so that a linear method can separate the data into the desirable classes. This approach is conceptually similar to the well-known method of support vector machines, which uses kernels to augment the dimensionality of the data, so that again only a simple linear method is sufficient to achieve data classification. In fact, a link between the purely statistical technique of support vector machines and the bio-inspired technique of reservoir computing has been formally built.[122] We can, perhaps, think of this link as a demonstration that biological inspiration and purely mathematical methodology might also solve problems in a similar manner.

We claim that reservoir computing would benefit from appropriate hardware. When simulating, the convergence of the recurrent network requires time, because the continuous system will be discretized and sequentially run on the central processing unit. If instead we replace the reservoir with an appropriate material, this step could become both fast and energy efficient: the material could compute effortlessly using its physical properties. Reservoirs do not need overengineering, because no specific structure is required; we only need to produce dynamics that are complex enough but not chaotic. In fact, there has already been work exploiting memristors in this direction.[123]

Could ideas from biology still add value to existing methods? A recent augmentation of the echo state networks,[124] inspired by the fruit fly brain, explores the concept of sparseness to improve learning performance of reservoirs. In brains, contrary to the typical ANNs, only few neurons fire at a time, a fact that has been linked to memory capacity. Neuronal thresholds appropriately initialized and updated with a slower time constant than that of the feedforward learnable weights can modulate

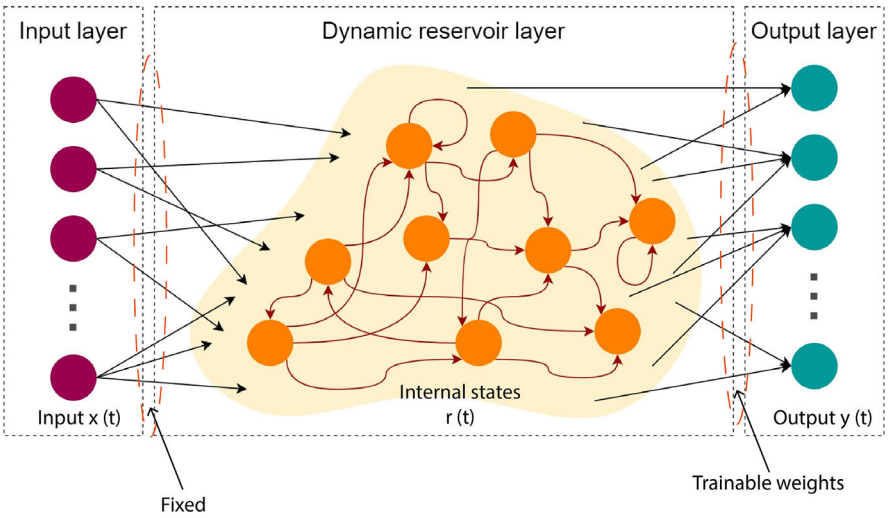


Figure 13. Reservoir computing maps inputs $x(t)$ to higher dimensional space, defined by the reservoir states $r(t)$. Only weights connecting reservoir states $r(t)$ and output $y(t)$ need to be trained.

sparseness and lead to better performance in comparison with the non-sparse reservoir, but also in comparison with state-of-the-art methods in a set of benchmark problems. Due to the sparseness leading to task-specific neurons, this bio-inspired technique can alleviate the problem of catastrophic forgetting. ML methods often suffer from the fact that once they learn a new task, they have forgotten the previous one. As in the space reservoir network, a new task will likely recruit previously unused neurons, and learning a new skill does not completely override those previously learned. This simple method competes and surpasses more complicated methods that are built specifically to address catastrophic forgetting. Most importantly, the formulation of the specific rule allows for completely replacing the network dynamics with any other dynamics, including material dynamics, that are suitable for the purpose (i.e., highly non-linear and not chaotic). Perhaps, there are more such lessons to be learned from biology.

So, what can be done right now? To us, it is clear that a better understanding of the physics behind memristive devices is key for the progress of the field.[125] A deeper understanding will allow us to harness the properties of the system for brain-like computation rather trying to fabricate some arbitrary brain behavior that may or may not be important in the context of a specific application, or worse may not scale up. Instead of thinking at the level of mimicking neurons and synapses, we can instead take inspiration from the biological systems, consider the dynamics required for neuronal processing, and use the material physics to reproduce them.

8. Conclusions

Memristor technologies are still to realize their full potential that has been promoted over the last 15 years. Although predominantly seen as candidates to replace or augment our current digital memory technologies, the impact of memristor technologies on the broader fields of AI and cognitive computing platforms is likely to be even more significant. As discussed in this progress report, the versatility of memristor technologies has resulted in their use across a range of applications: from in-memory computing, DL accelerators, and SNNs, to more futuristic bio-inspired computing paradigms. These approaches should not be seen as solutions to the same problem, nor as technologies that are in direct competition among themselves or with current, very successful, CMOS systems. In addition, it is crucial to recognize that many of the discussed research areas are still at the very beginning of their development. Of these, more mature approaches will likely produce industrially relevant solutions sooner. For example, greater power efficiency is an essential utility and a pressing issue that many engineers are trying to address. In-memory computing and DL accelerators based on memristors represent an attractive proposition for extreme power efficiency.

There is also significant scope for more fundamental work. Development of new generations of bio-inspired algorithms would further boost advancements in hardware systems and platforms. The challenge and opportunity lie in the interdisciplinary nature of the research and the necessity to understand distinct methodologies and approaches. We believe that the community will benefit from the next generation of researchers being well educated across different traditional disciplines. For example, there is an undeniable link between the fields of computer science, more specifically, ML, and computational neuroscience. The two disciplines could co-exist separately and act independently with distinct goals; however, there are great benefits to be gained from a more holistic approach. A strong case for closer collaboration has been made recently.[126] Collaborations should be expanded to include researchers in solid-state physics, materials science, nanoelectronics, circuit/architecture design, and information theory. Memristors show great promise to be a fabric for producing brain-inspired building blocks,[127] and this progress report showcases different types of memristor-based applications. Memristor technologies are versatile enough to

provide the perfect platform for different disciplines to strive together in pushing the frontiers of our current technologies in the most fundamental way. There are many specifics around memristive technologies, which have not been covered in this progress report. For example, optical control of memristive devices could potentially bring additional benefits in terms of higher bandwidth, lower cross talk, faster operational speeds, and integrate sensing together with memory and processing.[128–131] Integration of multiple functionalities in a compact nanodevice could lead to even better power efficiency of neuromorphic systems, such as artificial retinas.[132]

The progress report presents a broader landscape of ways memristors could be utilized for future computing systems. We aim to provide a general overview of the main approaches currently being pursued. In addition, the report provides some speculations about what might be missing in the research field and what efforts could be fruitful for the future. As such, the report does not include all details of different memristor technologies, materials systems, or specific technology challenges. There is a number of excellent review articles that cover specifics in much more details, and we refer readers to those. More specifically, an excellent overview of memristor-based electronics that discussed future prospects and current challenges can be found in the previous study.[7] More details about in-memory computing, including digital and analog schemes, are covered in the previous study.[16] The use of PCMs for brain-inspired computing is specifically discussed in the previous study,[23] whereas the use of redox-based memristors can be found in another review.[133] Other types of memristive technologies, not mentioned in this progress report, including ferroelectric memories, non-filamentary resistive random-access memory, and topological insulators, are covered in the recent guest editorial.[134] Integration of CMOS and memristive technologies for neuromorphic applications is discussed in the previous study.[118]

Acknowledgements

A.M. acknowledges funding and support from the Royal Academy of Engineering under the Research Fellowship scheme. A.S. acknowledges funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement number 682675). B.R. acknowledges partial support from IBM and Cisco. O.S. acknowledges funding from the European Research Council (ERC) under the European Union Horizon 2020 research and innovation program (grant agreement 725731). E.V. would like to acknowledge a Google Faculty Research Award (2017). A.J.K. acknowledges funding from the Engineering and Physical Sciences Research Council (EPSRC).

Conflict of Interest

The authors declare no conflict of interest.

Keywords

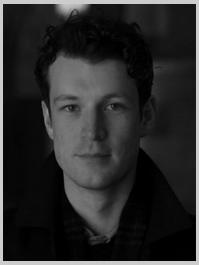
deep learning, in-memory computing, memristors, neuromorphic systems, power-efficient artificial intelligence, spiking neural networks

Received: April 23, 2020
Revised: June 22, 2020
Published online:

[1] D. Amodei, D. Hernandez, AI and Compute, https://openai.com/blog/ai-and-compute/ (accessed: March 2020).
[2] M. M. Waldrop, *Nature* **2016**, *530*, 144.
[3] V. Sze, Y.-H. Chen, T.-J. Yang, J. S. Emer, *Proc. IEEE* **2017**, *105*, 2295.
[4] *Resistive Switching: From Fundamentals of Nanoionic Redox Processes to Memristive Device Applications* (Eds: D. Ielmini, R. Waser), Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim **2016**.
[5] D. B. Strukov, G. S. Snider, D. R. Stewart, R. S. Williams, *Nature* **2008**, *453*, 80.
[6] K. Szot, W. Speier, G. Bihlmayer, R. Waser, *Nat. Mater* **2006**, *5*, 312.
[7] M. A. Zidan, J. P. Strachan, W. D. Lu, *Nat. Electron.* **2018**, *1*, 22.
[8] L. Chua, *IEEE Trans. Circuit Theory* **1971**, *18*, 507.
[9] A. Mehonic, A. J. Kenyon, in *Defects at Oxide Surfaces* (Eds.: J. Jupille, G. Thornton), Springer International Publishing, Cham **2015**, pp. 401–428.
[10] S. Yu, *Neuro-Inspired Computing Using Resistive Synaptic Devices*, Springer Science+Business Media, New York, NY **2017**.
[11] O. Mutlu, S. Ghose, J. Gómez-Luna, R. Ausavarungnirun, *Microprocess. Microsyst.* **2019**, *67*, 28.
[12] S. W. Keckler, W. J. Dally, B. Khailany, M. Garland, D. Glasco, *IEEE Micro.* **2011**, *31*, 7.
[13] N. P. Jouppi, A. Borchers, R. Boyle, P. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, C. Young, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, N. Patil, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, et al., in *Proc. 44th Annual Int. Symp. Computer Architecture – ISCA '17*, ACM Press, Toronto, ON, Canada **2017**, pp. 1–12.
[14] A. Sebastian, T. Tuma, N. Papandreou, M. Le Gallo, L. Kull, T. Parnell, E. Eleftheriou, *Nat. Commun.* **2017**, *8*, 1115.
[15] J. J. Yang, D. B. Strukov, D. R. Stewart, *Nat. Nanotechnol.* **2013**, *8*, 13.
[16] D. Ielmini, H.-S. P. Wong, *Nat. Electron.* **2018**, *1*, 333.
[17] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, E. Eleftheriou, *Nat. Nanotechnol.* **2020**, *15*, 529.
[18] M. Di Ventra, Y. V. Pershin, *Nat. Phys.* **2013**, *9*, 200.
[19] Z. Sun, G. Pedretti, E. Ambrosi, A. Bricalli, W. Wang, D. Ielmini, *Proc. Natl. Acad. Sci. USA* **2019**, *116*, 4123.
[20] L. Chua, *Appl. Phys. A* **2011**, *102*, 765.
[21] H.-S. P. Wong, S. Salahuddin, *Nat. Nanotechnol.* **2015**, *10*, 191.
[22] A. Sebastian, M. Le Gallo, E. Eleftheriou, *J. Phys. D: Appl. Phys.* **2019**, *52*, 443002.
[23] A. Sebastian, M. Le Gallo, G. W. Burr, S. Kim, M. BrightSky, E. Eleftheriou, *J. Appl. Phys.* **2018**, *124*, 111101.
[24] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, R. S. Williams, *Nature* **2010**, *464*, 873.
[25] I. Vourkas, G. Ch. Sirakoulis, *IEEE Circuits Syst. Mag.* **2016**, *16*, 15.
[26] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, U. C. Weiser, *IEEE Trans. Circuits Syst. II* **2014**, *61*, 895.
[27] A. Haj-Ali, R. Ben-Hur, N. Wald, R. Ronen, S. Kvatinsky, *IEEE Trans. Circuits Syst. I* **2018**, *65*, 4258.
[28] S. Hamdioui, H. A. Du Nguyen, M. Taouil, A. Sebastian, M. L. Gallo, S. Pande, S. Schaafsma, F. Catthoor, S. Das, F. G. Redondo, G. Karunaratne, A. Rahimi, L. Benini, in *2019 Design, Automation & Test in Europe Conf. Exhibition (DATE)*, IEEE, Florence, Italy **2019**, pp. 486–491.

[29] A. Rahimi, S. Datta, D. Kleyko, E. P. Frady, B. Olshausen, P. Kanerva, J. M. Rabaey, *IEEE Trans. Circuits Syst. I* **2017**, *64*, 2508.
[30] G. Karunaratne, M. Le Gallo, G. Cherubini, L. Benini, A. Rahimi, A. Sebastian, *Nat. Electron.* **2020**, *3*, 327.
[31] N. Papandreou, H. Pozidis, A. Pantazi, A. Sebastian, M. Breitwisch, C. Lam, E. Eleftheriou, in *2011 IEEE Int. Symp. of Circuits and Systems (ISCAS)*, IEEE, Rio de Janeiro, Brazil **2011**, pp. 329–332.
[32] R. Carboni, D. Ielmini, *Adv. Electron. Mater.* **2019**, *5*, 1900198.
[33] Y. Shim, S. Chen, A. Sengupta, K. Roy, *Sci. Rep.* **2017**, *7*, 14101.
[34] H. Nili, G. C. Adam, B. Hoskins, M. Prezioso, J. Kim, M. R. Mahmoodi, F. M. Bayat, O. Kavehei, D. B. Strukov, *Nat. Electron.* **2018**, *1*, 197.
[35] G. W. Burr, R. M. Shelby, A. Sebastian, S. Kim, S. Kim, S. Sidler, K. Virwani, M. Ishii, P. Narayanan, A. Fumarola, L. L. Sanches, I. Boybat, M. Le Gallo, K. Moon, J. Woo, H. Hwang, Y. Leblebici, *Adv. Phys. X* **2017**, *2*, 89.
[36] M. Le Gallo, A. Sebastian, G. Cherubini, H. Giefers, E. Eleftheriou, *IEEE Trans. Electron Devices* **2018**, *65*, 4304.
[37] B. Fleischer, S. Shukla, M. Ziegler, J. Silberman, J. Oh, V. Srinivasan, J. Choi, S. Mueller, A. Agrawal, T. Babinsky, N. Cao, C.-Y. Chen, P. Chuang, T. Fox, G. Gristede, M. Guillorn, H. Haynie, M. Klaiber, D. Lee, S.-H. Lo, G. Maier, M. Scheuermann, S. Venkataramani, C. Vezyrtzis, N. Wang, F. Yee, C. Zhou, P.-F. Lu, B. Curran, L. Chang, K. Gopalakrishnan, in *2018 IEEE Symp. VLSI Circuits*, IEEE, Honolulu, HI **2018**, pp. 35–36.
[38] G. W. Burr, R. M. Shelby, S. Sidler, C. di Nolfo, J. Jang, I. Boybat, R. S. Shenoy, P. Narayanan, K. Virwani, E. U. Giacometti, B. N. Kurdi, H. Hwang, *IEEE Trans. Electron Devices* **2015**, *62*, 3498.
[39] V. Joshi, M. Le Gallo, S. Haefeli, I. Boybat, S. R. Nandakumar, C. Piveteau, M. Dazzi, B. Rajendran, A. Sebastian, E. Eleftheriou, *Nat. Commun.* **2020**, *11*, 2473.
[40] A. Mehonic, D. Joksas, W. H. Ng, M. Buckwell, A. J. Kenyon, *Front. Neurosci.* **2019**, *13*, 593.
[41] M. Le Gallo, D. Krebs, F. Zipoli, M. Salinga, A. Sebastian, *Adv. Electron. Mater.* **2018**, *4*, 1700627.
[42] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio, *J. Mach. Learn. Res.* **2017**, *18*, 6869.
[43] M. Le Gallo, A. Sebastian, R. Mathis, M. Manica, H. Giefers, T. Tuma, C. Bekas, A. Curioni, E. Eleftheriou, *Nat. Electron.* **2018**, *1*, 246.
[44] S. R. Nandakumar, M. Le Gallo, C. Piveteau, V. Joshi, G. Mariani, I. Boybat, G. Karunaratne, R. Khaddam-Aljameh, U. Egger, A. Petropoulos, T. Antonakopoulos, B. Rajendran, A. Sebastian, E. Eleftheriou, *Front. Neurosci.* **2020**, *14*, 406.
[45] S. R. Nandakumar, M. Le Gallo, I. Boybat, B. Rajendran, A. Sebastian, E. Eleftheriou, in *2018 IEEE Int. Symp. on Circuits and Systems (ISCAS)*, IEEE, Florence **2018**, pp. 1–5.
[46] E. Eleftheriou, M. L. Gallo, S. R. Nandakumar, C. Piveteau, I. Boybat, V. Joshi, R. Khaddam-Aljameh, M. Dazzi, I. Giannopoulos, G. Karunaratne, B. Kersting, M. Stanisavljevic, V. P. Jonnalagadda, N. Ioannou, K. Kourtis, P. A. Francese, A. Sebastian, *IBM J. Res. Dev.* **2019**, *63*, 1.
[47] F. Alibart, E. Zamanidoost, D. B. Strukov, *Nat. Commun.* **2013**, *4*, 2072.
[48] T. Gokmen, Y. Vlasov, *Front. Neurosci.* **2016**, *10*, 333.
[49] S. Ambrogio, P. Narayanan, H. Tsai, R. M. Shelby, I. Boybat, C. di Nolfo, S. Sidler, M. Giordano, M. Bodini, N. C. P. Farinha, B. Killeen, C. Cheng, Y. Jaoudi, G. W. Burr, *Nature* **2018**, *558*, 60.
[50] V. Seshadri, T. C. Mowry, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M. A. Kozuch, O. Mutlu, P. B. Gibbons, in *Proc. of the 50th Annual IEEE/ACM Int. Symp. on Microarchitecture – MICRO-50 '17*, ACM Press, Cambridge, MA **2017**, pp. 273–287.
[51] S. Aga, S. Jeloka, A. Subramaniyan, S. Narayanasamy, D. Blaauw, R. Das, in *2017 IEEE Int. Symp. on High Performance Computer Architecture (HPCA)*, IEEE, Austin, TX **2017**, pp. 481–492.
[52] N. Verma, H. Jia, H. Valavi, Y. Tang, M. Ozatay, L.-Y. Chen, B. Zhang, P. Deaville, *IEEE Solid-State Circuits Mag.* **2019**, *11*, 43.
[53] F. Xiong, A. D. Liao, D. Estrada, E. Pop, *Science* **2011**, *332*, 568.
[54] K.-S. Li, C. Ho, M.-T. Lee, M.-C. Chen, C.-L. Hsu, J. M. Lu, C. H. Lin, C. C. Chen, B. W. Wu, Y. F. Hou, C. Yi. Lin, Y. J. Chen, T. Y. Lai, M. Y. Li, I. Yang, C. S. Wu, F.-L. Yang, in *2014 Symp. on VLSI Technology (VLSI-Technology): Digest of Technical Papers*, IEEE, Honolulu, HI **2014**, pp. 1–2.
[55] M. Salinga, B. Kersting, I. Ronneberger, V. P. Jonnalagadda, X. T. Vu, M. Le Gallo, I. Giannopoulos, O. Cojocaru-Mirédin, R. Mazzarello, A. Sebastian, *Nat. Mater* **2018**, *17*, 681.
[56] S. Pi, C. Li, H. Jiang, W. Xia, H. Xin, J. J. Yang, Q. Xia, *Nature Nanotechnol.* **2019**, *14*, 35.
[57] M. Buckwell, L. Montesi, S. Hudziak, A. Mehonic, A. J. Kenyon, *Nanoscale* **2015**, *7*, 18030.
[58] S. Brivio, J. Frascaroli, S. Spiga, *Nanotechnology* **2017**, *28*, 395202.
[59] S. Choi, S. H. Tan, Z. Li, Y. Kim, C. Choi, P.-Y. Chen, H. Yeon, S. Yu, J. Kim, *Nat. Mater* **2018**, *17*, 335.
[60] I. Boybat, M. Le Gallo, S. R. Nandakumar, T. Moraitis, T. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, E. Eleftheriou, *Nat. Commun.* **2018**, *9*, 2514.
[61] W. W. Koelmans, A. Sebastian, V. P. Jonnalagadda, D. Krebs, L. Dellmann, E. Eleftheriou, *Nat. Commun.* **2015**, *6*, 8181.
[62] I. Giannopoulos, A. Sebastian, M. Le Gallo, V. P. Jonnalagadda, M. Sousa, M. N. Boon, E. Eleftheriou, in *2018 IEEE International Electron Devices Meeting (IEDM)*, IEEE, San Francisco, CA **2018**, pp. 27.7.1–27.7.4.
[63] S. Yu, *Proc. IEEE* **2018**, *106*, 260.
[64] M. Le Gallo, T. Tuma, F. Zipoli, A. Sebastian, E. Eleftheriou, in *2016 46th European Solid-State Device Research Conf. (ESSDERC)*, IEEE, Lausanne, Switzerland **2016**, pp. 373–376.
[65] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, H.-S. P. Wong, in *2012 Int. Electron Devices Meeting*, IEEE, San Francisco, CA **2012**, pp. 10.4.1–10.4.4.
[66] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, H. Qian, *Nature* **2020**, *577*, 641.
[67] 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, D. S. Modha, *Science* **2014**, *345*, 668.
[68] D. Kuzum, R. G. D. Jeyasingh, B. Lee, H.-S. P. Wong, *Nano Lett.* **2012**, *12*, 2179.
[69] M. Jerry, W. Tsai, B. Xie, X. Li, V. Narayanan, A. Raychowdhury, S. Datta, in *2016 74th Annual Device Research Conference (DRC)*, IEEE, Newark, DE **2016**, pp. 1–2.
[70] T. Tuma, A. Pantazi, M. L. Gallo, A. Sebastian, E. Eleftheriou, *Nat. Nanotechnol.* **2016**, *11*, 693.
[71] A. Mehonic, A. J. Kenyon, *Front. Neurosci.* **2016**, *10*, 57.
[72] A. Sengupta, A. Ankit, K. Roy, in *2017 Int. Joint Conf. on Neural Networks (IJCNN)*, IEEE, Anchorage, AK **2017**, pp. 4557–4563.
[73] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, M. Pfeiffer, in *2015 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Killarney, Ireland **2015**, pp. 1–8.
[74] A. Sengupta, Y. Ye, R. Wang, C. Liu, K. Roy, *Front. Neurosci.* **2019**, *13*, 95.
[75] R. Midya, Z. Wang, S. Asapu, S. Joshi, Y. Li, Y. Zhuo, W. Song, H. Jiang, N. Upadhay, M. Rao, P. Lin, C. Li, Q. Xia, J. J. Yang, *Adv. Electron. Mater.* **2019**, *5*, 1900060.
[76] G. Bi, M. Poo, *J. Neurosci.* **1998**, *18*, 10464.

[77] H. Shouval, S. S.-H. Wang, G. M. Wittenberg, *Front. Comput. Neurosci.* **2010**, *4*, 19.
[78] Z. Brzosko, S. B. Mierau, O. Paulsen, *Neuron* **2019**, *103*, 563.
[79] J. Seo, B. Brezzo, Y. Liu, B. D. Parker, S. K. Esser, R. K. Montoye, B. Rajendran, J. A. Tierno, L. Chang, D. S. Modha, D. J. Friedman, in *2011 IEEE Custom Integrated Circuits Conference (CICC)*, IEEE, San Jose, CA **2011**, pp. 1–4.
[80] S. Kim, C. Du, P. Sheridan, W. Ma, S. Choi, W. D. Lu, *Nano Lett.* **2015**, *15*, 2203.
[81] K. Zarudnyi, A. Mehonic, L. Montesi, M. Buckwell, S. Hudziak, A. J. Kenyon, *Front. Neurosci.* **2018**, *12*, 57.
[82] S. Kim, M. Ishii, S. Lewis, T. Perri, M. BrightSky, W. Kim, R. Jordan, G. W. Burr, N. Sosa, A. Ray, J.-P. Han, C. Miller, K. Hosokawa, C. Lam, in *2015 IEEE International Electron Devices Meeting (IEDM)*, IEEE, Washington, DC **2015**, pp. 17.1.1–17.1.4.
[83] A. Serb, J. Bill, A. Khiat, R. Berdan, R. Legenstein, T. Prodromakis, *Nat. Commun.* **2016**, *7*, 12611.
[84] Y. Fang, Z. Wang, J. Gomez, S. Datta, A. I. Khan, A. Raychowdhury, *Front. Neurosci.* **2019**, *13*, 855.
[85] N. Anwani, B. Rajendran, *Neurocomputing* **2020**, *380*, 67.
[86] F. Zenke, S. Ganguli, *Neural Comput.* **2018**, *30*, 1514.
[87] O. Neftci, H. Mostafa, F. Zenke, *IEEE Signal Process. Mag.* **2019**, *36*, 51.
[88] S. R. Nandakumar, I. Boybat, M. L. Gallo, E. Eleftheriou, A. Sebastian, B. Rajendran, *Sci. Rep.* **2020**, *10*, 8080.
[89] J. Wouters, Y.-Y. Chen, A. Fantini, N. Raghavan, in *Resistive Switching* (Eds.: D. Ielmini, R. Waser), Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, Germany **2016**, pp. 597–622.
[90] C. Sung, S. Lim, H. Kim, T. Kim, K. Moon, J. Song, J.-J. Kim, H. Hwang, *Nanotechnology* **2018**, *29*, 115203.
[91] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves, Z. Li, J. P. Strachan, P. Lin, Z. Wang, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, Q. Xia, *Nat. Electron.* **2018**, *1*, 52.
[92] Y. Fang, Z. Yu, Z. Wang, T. Zhang, Y. Yang, Y. Cai, R. Huang, *IEEE Electron Device Lett.* **2018**, *39*, 819.
[93] J. Kenyon, M. Singh Munde, W. H. Ng, M. Buckwell, D. Joksas, A. Mehonic, *Faraday Discuss.* **2019**, *213*, 151.
[94] S. Munde, A. Mehonic, W. H. Ng, M. Buckwell, L. Montesi, M. Bosman, A. L. Shluger, A. J. Kenyon, *Sci. Rep.* **2017**, *7*, 9274.
[95] A. Mehonic, M. S. Munde, W. H. Ng, M. Buckwell, L. Montesi, M. Bosman, A. L. Shluger, A. J. Kenyon, *Microelectron. Eng.* **2017**, *178*, 98.
[96] L. Xia, W. Huangfu, T. Tang, X. Yin, K. Chakrabarty, Y. Xie, Y. Wang, H. Yang, *IEEE J. Emerg. Sel. Topics Circuits Syst.* **2018**, *8*, 102.
[97] M. Hu, J. P. Strachan, Z. Li, R. S. Williams, in *2016 17th Int. Symp. on Quality Electronic Design (ISQED)*, IEEE, Santa Clara, CA **2016**, pp. 374–379.
[98] D. Joksas, P. Freitas, Z. Chai, W. H. Ng, M. Buckwell, W. D. Zhang, A. J. Kenyon, A. Mehonic, *arXiv:1909.06658 [cs]* **2019**.
[99] S. Park, A. Sheri, J. Kim, J. Noh, J. Jang, M. Jeon, B. Lee, B. R. Lee, B. H. Lee, H. Hwang, in *2013 IEEE Int. Electron Devices Meeting*, IEEE, Washington, DC **2013**, pp. 25.6.1–25.6.4.
[100] I.-T. Wang, C.-C. Chang, L.-W. Chiu, T. Chou, T.-H. Hou, *Nanotechnology* **2016**, *27*, 365204.
[101] J. Woo, K. Moon, J. Song, S. Lee, M. Kwak, J. Park, H. Hwang, *IEEE Electron Device Lett.* **2016**, *37*, 994.
[102] W. Maass, *Proc. IEEE* **2014**, *102*, 860.
[103] M. Payvand, M. V. Nair, L. K. Müller, G. Indiveri, *Faraday Discuss.* **2019**, *213*, 487.
[104] M. Al-Shedivat, R. Naous, G. Cauwenberghs, K. N. Salama, *IEEE J. Emerg. Sel. Topics Circuits Syst.* **2015**, *5*, 242.
[105] P. Shukla, A. Shylendra, T. Tulabandhula, A. R. Trivedi, *arXiv:2003.02629 [cs, eess]* **2020**.
[106] K. Yang, A. Malhotra, S. Lu, A. Sengupta, *IEEE Trans. Electron Devices* **2020**, *67*, 1340.
[107] A. Malhotra, S. Lu, K. Yang, A. Sengupta, *IEEE Trans. Nanotechnol.* **2020**, *19*, 328.
[108] N. Wycoff, P. Balaprakash, F. Xia, *arXiv:2005.04165 [cs, stat]* **2020**.
[109] J. Brea, W. Senn, J.-P. Pfister, *J. Neurosci.* **2013**, *33*, 9565.
[110] B. Rosenfeld, O. Simeone, B. Rajendran, in *2019 IEEE 20th Int. Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, IEEE, Cannes, France **2019**, pp. 1–5.
[111] H. Jang, O. Simeone, in *ICASSP 2019–2019 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, Brighton, UK **2019**, pp. 3382–3386.
[112] S.-C. Liu, B. Rueckauer, E. Ceolini, A. Huber, T. Delbruck, *IEEE Signal Process. Mag.* **2019**, *36*, 29.
[113] A. Bagheri, O. Simeone, B. Rajendran, in *2018 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, Calgary, AB **2018**, pp. 2986–2990.
[114] E. Rumelhart, G. E. Hinton, R. J. Williams, *Nature* **1986**, *323*, 533.
[115] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, D. Hassabis, *Nature* **2017**, *550*, 354.
[116] E. Strubell, A. Ganesh, A. McCallum, in *Proc. of the 57th Annu. Meeting of the Association for Computational Linguistics*, Association For Computational Linguistics, Florence, Italy **2019**, pp. 3645–3650.
[117] Q. Xia, J. J. Yang, *Nat. Mater.* **2019**, *18*, 309.
[118] M. Rahimi Azghadi, Y.-C. Chen, J. K. Eshraghian, J. Chen, C.-Y. Lin, A. Amirsoleimani, A. Mehonic, A. J. Kenyon, B. Fowler, J. C. Lee, Y.-F. Chang, *Adv. Intell. Syst.* **2020**, *2*, 1900189.
[119] H. Jaeger, *The "ECHO STATE" Approach to Analysing and Training Recurrent Neural Networks*, GMD – German National Research Institute For Computer Science, Bonn, Germany **2001**.
[120] W. Maass, T. Natschläger, H. Markram, *Neural Comput.* **2002**, *14*, 2531.
[121] L. Manneschi, E. Vasilaki, *Nat. Mach. Intell.* **2020**, *2*, 155.
[122] M. Hermans, B. Schrauwen, *Neural Comput.* **2012**, *24*, 104.
[123] C. Du, F. Cai, M. A. Zidan, W. Ma, S. H. Lee, W. D. Lu, *Nat. Commun.* **2017**, *8*, 2204.
[124] L. Manneschi, A. C. Lin, E. Vasilaki, *arXiv:1912.08124 [cs, stat]* **2019**.
[125] M. Lanza, H.-S. P. Wong, E. Pop, D. Ielmini, D. Strukov, B. C. Regan, L. Larcher, M. A. Villena, J. J. Yang, L. Goux, A. Belmonte, Y. Yang, F. M. Puglisi, J. Kang, B. Magyari-Köpe, E. Yalon, A. Kenyon, M. Buckwell, A. Mehonic, A. Shluger, H. Li, T.-H. Hou, B. Hudec, D. Akinwande, R. Ge, S. Ambrogio, J. B. Roldan, E. Miranda, J. Suñe, K. L. Pey, et al., *Adv. Electron. Mater.* **2019**, *5*, 1800143.
[126] B. Aimone, *Commun. ACM* **2019**, *62*, 110.
[127] D. Kendall, S. Kumar, *Appl. Phys. Rev.* **2020**, *7*, 011305.
[128] F. Zhou, Z. Zhou, J. Chen, T. H. Choy, J. Wang, N. Zhang, Z. Lin, S. Yu, J. Kang, H.-S. P. Wong, Y. Chai, *Nat. Nanotechnol.* **2019**, *14*, 776.
[129] H. Tan, G. Liu, X. Zhu, H. Yang, B. Chen, X. Chen, J. Shang, W. D. Lu, Y. Wu, R.-W. Li, *Adv. Mater.* **2015**, *27*, 2797.
[130] A. Mehonic, T. Gerard, A. J. Kenyon, *Appl. Phys. Lett.* **2017**, *111*, 233502.
[131] W. Xue, W. Ci, X.-H. Xu, G. Liu, *Chinese Phys. B* **2020**, *29*, 048401.
[132] L. Bao, J. Kang, Y. Fang, Z. Yu, Z. Wang, Y. Yang, Y. Cai, R. Huang, *Sci. Rep.* **2018**, *8*, 13727.
[133] R. Dittmann, J. P. Strachan, *APL Mater.* **2019**, *7*, 110903.
[134] W. Burr, A. Sebastian, E. Vianello, R. Waser, S. Parkin, *APL Mater.* **2020**, *8*, 010401.



Adnan Mehonic (Ph.D. in electronic engineering at UCL, UK, 2014) is a lecturer in nanoelectronics and a Royal Academy of Engineering Research fellow in the Department of Electrical and Electronic Engineering, UCL. He works on the development of power-efficient computing systems (neuromorphic systems) based on memristors. The work includes co-design of devices, circuits, and algorithms that would enable on-chip implementation of ML/AI. He is interested in both accelerators for conventional ML and non-conventional methods for information processing (e.g., spike-based computing). He is a co-founder of Intrinsic Semiconductor Technologies.

Anthony J. Kenyon is a professor of nanoelectronic & nanophotonic materials and a vice dean (research) in the Faculty of Engineering Sciences at UCL. He is a fellow of the Institute of Physics and of the IET, a Senior Member of the IEEE, and a member of the IEEE Electron Devices Society, IEEE Nanotechnology Council, and the Executive Committee of the European Materials Research Society (E-MRS). His group's work focuses on the application of nanostructured materials to photonics and nanoelectronics. He is a co-founder of Intrinsic Semiconductor Technologies, a spin-out company setup to commercialize silicon oxide memristive technologies.