Tutorial: Brain-inspired computing using phase-change memory devices

Abu Sebastian,[1] Manuel Le Gallo,[1] Geoffrey W. Burr,[2] Sangbum Kim,[3] Matthew BrightSky,[3] and Evangelos Eleftheriou[1]
[1]*IBM Research–Zurich, Säumerstrasse 4, 8803 Rüschlikon, Switzerland*
[2]*IBM Research–Almaden, 650 Harry Road, San Jose, California 95120, USA*
[3]*IBM T. J. Watson Research Center, 1101 Kitchawan Road, Yorktown Heights, New York 10598, USA*

(Received 31 May 2018; accepted 20 August 2018; published online 18 September 2018)

There is a significant need to build efficient non-von Neumann computing systems for highly data-centric artificial intelligence related applications. Brain-inspired computing is one such approach that shows significant promise. Memory is expected to play a key role in this form of computing and, in particular, phase-change memory (PCM), arguably the most advanced emerging non-volatile memory technology. Given a lack of comprehensive understanding of the working principles of the brain, brain-inspired computing is likely to be realized in multiple levels of inspiration. In the first level of inspiration, the idea would be to build computing units where memory and processing co-exist in some form. Computational memory is an example where the physical attributes and the state dynamics of memory devices are exploited to perform certain computational tasks in the memory itself with very high areal and energy efficiency. In a second level of brain-inspired computing using PCM devices, one could design a co-processor comprising multiple cross-bar arrays of PCM devices to accelerate the training of deep neural networks. PCM technology could also play a key role in the space of specialized computing substrates for spiking neural networks, and this can be viewed as the third level of brain-inspired computing using these devices. *© 2018 Author(s). All article content, except where otherwise noted, is licensed under a Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).* https://doi.org/10.1063/1.5042413

I. INTRODUCTION

We are on the cusp of a revolution in artificial intelligence (AI) and cognitive computing. The computing systems that run today's AI algorithms are based on the von Neumann architecture where large amounts of data need to be shuttled back and forth at high speeds during the execution of these computational tasks (see Fig. 1). This creates a performance bottleneck and also leads to significant area/power inefficiency. Thus, it is becoming increasingly clear that to build efficient cognitive computers, we need to transition to novel architectures where memory and processing are better collocated. Brain-inspired computing is a key non-von Neumann approach that is being actively researched. It is natural to be drawn to the human brain for inspiration, a remarkable engine of cognition that performs computation on the order of peta-ops per joule thus providing an "existence proof" for an ultralow power cognitive computer. Unfortunately, we are still quite far from attaining a comprehensive understanding of how the brain computes. However, we have uncovered certain salient features of this computing system such as the collocation of memory and processing, a computing fabric comprising large-scale networks of neurons and plastic synapses and spike-based communication and processing of information. Based on these insights, we could begin to realize brain-inspired computing systems at multiple levels of inspiration or abstraction.

In the brain, memory and processing are highly entwined. Hence, the memory unit can be expected to play a key role in brain-inspired computing systems. In particular, very high-density, low-power, variable-state, programmable and non-volatile memory devices could play a central role. One such nanoscale memory device is phase-change memory (PCM).[1] PCM is based on the property of certain compounds of Ge, Te, and Sb that exhibit drastically different electrical characteristics depending on their atomic arrangement.[2] In the disordered amorphous phase, these materials have very high resistivity, while in the ordered crystalline phase, they have very low resistivity.

A PCM device consists of a nanometric volume of this phase change material sandwiched between two electrodes. A schematic illustration of a PCM device with a "mushroom-type" device geometry is shown in Fig. 2(a). The phase change material is in the crystalline phase in an as-fabricated device. In a memory array, the PCM devices are typically placed in series with an access device such as a field effect transistor (FET) referred to as a 1T1R configuration. When a current pulse of sufficiently high amplitude is applied to the PCM device (typically referred to as the RESET pulse), a significant portion of the phase change material melts owing to Joule heating. The typical melting temperature of phase-change materials is approx. 600 °C. When the pulse is stopped abruptly so that temperature inside the heated device drops rapidly, the molten material quenches into the amorphous phase due to glass transition. In the resulting RESET state, the device will be in a high resistance state if the amorphous region blocks the bottom electrode. A transmission electron micrograph of a PCM

 © Author(s) 2018.

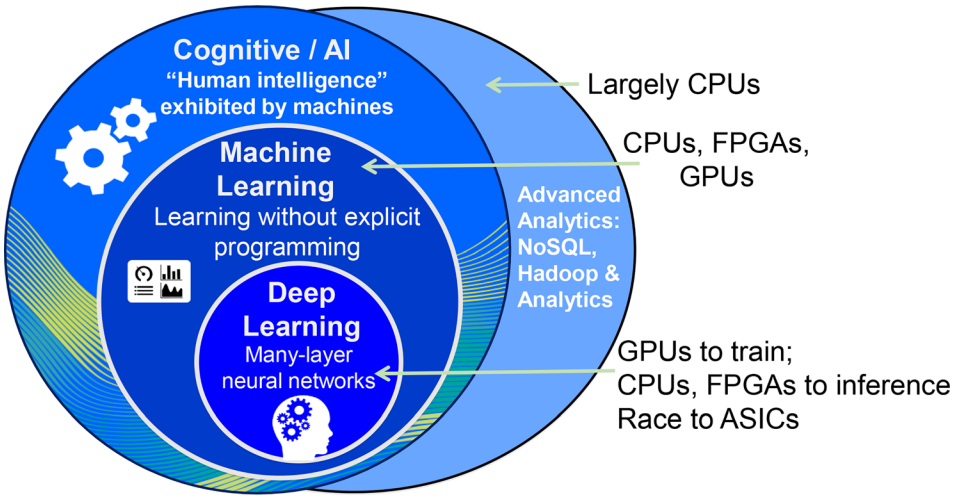

FIG. 1. Most of the algorithms related to artificial intelligence run on conventional computing systems such as central processing units (CPUs), graphical processing units (GPUs) and field programmable gate arrays (FPGAs). There is also significant recent interest in application specific integrated circuits (ASICs). In all these computing systems, the memory and processing units are physically separated and hence a significant amount of data need to be shuttled back and forth during computation. This creates a performance bottleneck.

device in the RESET state is shown in Fig. 2(b). When a current pulse (typically referred to as the SET pulse) is applied to a PCM device in the RESET state, such that the temperature reached in the cell via Joule heating is high, but below the melting temperature, a part of the amorphous region crystallizes. The temperature that corresponds to the highest rate of crystallization is typically ≈400 °C. In particular, if the SET pulse induces complete crystallization, then the device will be in a low resistance state. In this scenario, we have a memory device that can store one bit of information. The memory state can be read by biasing the device with a small amplitude read voltage that does not disturb the phase-configuration.

The first key property of PCM that enables brain-inspired computing is its ability to achieve not just two levels but a continuum of resistance or conductance values.[3] This is typically achieved by creating intermediate phase configurations by the application of suitable partial RESET pulses.[4,5] For example, in Fig. 3(a), it is shown how one can achieve a continuum of resistance levels by the application of RESET pulses with varying amplitude. The device is first programmed to the fully crystalline state. Thereafter, RESET pulses are applied with progressively increasing amplitude. The resistance is measured after the application of each RESET pulse. It can be seen that the device resistance, related to the size of the amorphous region (shown in red), increases with increasing RESET current. The curve shown in Fig. 3(a) is typically referred to as the programming curve. The programming curve is usually bidirectional (can increase as well as decrease the resistance by modulating the programming current) and is typically employed when one has to program a PCM device to a certain desired resistance value. This is achieved through iterative programming by applying several pulses in a closed-loop manner.[5] The programming curves are shown in terms of the programming current due to the highly nonlinear *I-V* characteristics of the PCM devices. A slight variation in the programming voltage would result in large variations in the programming current. For example, for the devices shown in Fig. 3, a voltage drop across the PCM devices of 1.0 V corresponds to 100 μA and 1.2 V corresponds to 500 μA. The latter results in a dissipated power of 600 μW and the energy expended assuming a pulse duration of 50 ns is 30 pJ. An additional consideration is that the amorphous phase-change material has to undergo

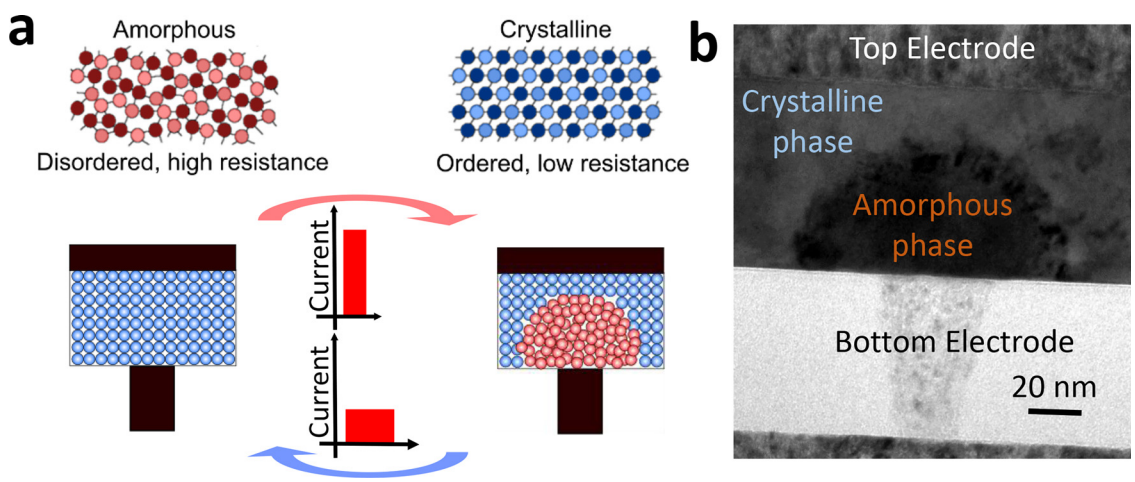

FIG. 2. (a) Phase-change memory is based on the rapid and reversible phase transition of certain types of materials between crystalline and amorphous phases by the application of suitable electrical pulses. (b) Transmission electron micrograph of a mushroom-type PCM device in a RESET state. It can be seen that the bottom electrode is blocked by the amorphous phase.

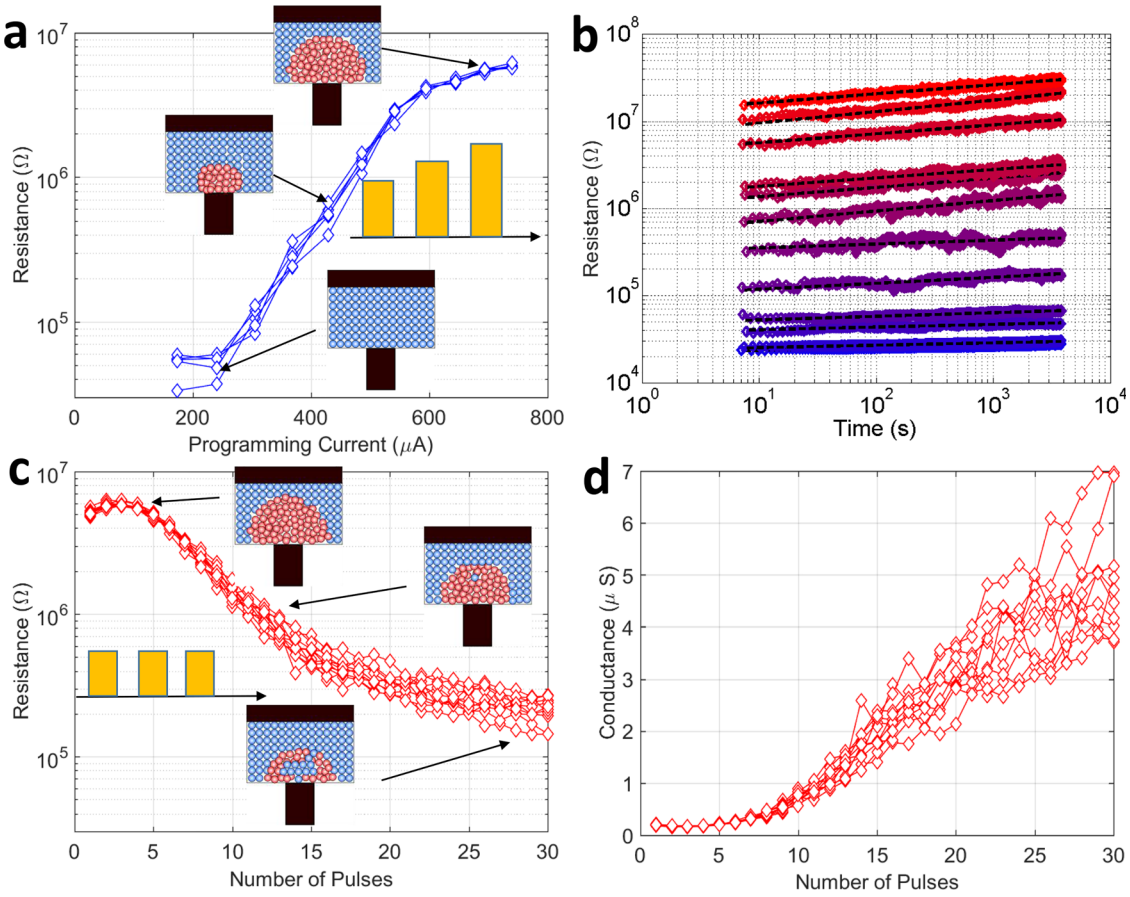

FIG. 3. (a) Characteristic programming curves that show the achievable resistance values as a function of partial RESET pulses with varying amplitude. This bi-directional curve is typically employed to achieve a desired resistance value using iterative programming. (b) Temporal fluctuations associated with the resistance values due to structural relaxation of the amorphous phase as well as 1/*f* noise. (c) Characteristic accumulation curves that show the evolution of resistance values as a function of the successive application of a SET pulse with the same amplitude. Due to the crystallization dynamics, there is a progressive reduction in the size of the amorphous region and the resulting device resistance. (d) Illustration of the accumulative behavior where the progressive increase in device conductance is depicted in a linear scale.

threshold switching prior to being able to conduct such high currents at such low voltage values.[6,7] This could necessitate voltage values of up to 2.5 V.

Even though it is possible to achieve a desired resistance value through iterative programming, there are significant temporal fluctuations associated with the resistance values [see Fig. 3(b)]. For example, PCM devices exhibit significant 1/*f* noise behavior.[8] There is also a temporal evolution of resistance arising from a spontaneous structural relaxation of the amorphous phase.[9,10] The thermally activated nature of electrical transport also leads to significant resistance changes resulting from ambient temperature variations.[11]

The second key property that enables brain-inspired computing is the accumulative behavior arising from the crystallization dynamics.[12] As shown in Fig. 3(c), one can induce progressive reduction in the size of the amorphous region (and hence the device resistance) by the successive application of SET pulses with the same amplitude. However, it is not possible to achieve a progressive increase in the size of the amorphous region. Hence, the curve shown in Fig. 3(c) typically referred to as the accumulation curve, is unidirectional. The SET pulses typically consume less energy (approx. 5 pJ) compared to the RESET pulses. As we will see later on, it is often desirable to achieve a linear increase in conductance as a function of the number of SET pulses. However, as shown in Fig. 3(d), this desired behavior is not what real devices tend to exhibit. It can also be seen that there is significant cycle-to-cycle randomness associated with the accumulation process attributed to the inherent stochasticity associated with the crystallization process.[13–15]

In Secs. II–IV, we will describe how the multi-level storage capability and the accumulative behavior can be exploited for brain-inspired computing.

II. COMPUTATIONAL MEMORY

At a basic level, a key attribute of brain-inspired computing is the co-location of memory and processing. It can be shown that it is possible to perform in-place computation with data stored in PCM devices. The essential idea is not to treat memory as a passive storage entity, but to exploit the physical attributes of the memory devices as described in Sec. I, and thus realize computation exactly at the place where the data are stored. We will refer to this first level of inspiration as in-memory computing and refer to the memory unit that performs in-memory computing as computational memory (see Fig. 4). Several computational tasks such as logical operations,[16] arithmetic operations[17,18] and even

certain machine learning tasks[19] can be implemented in such a computational memory unit.

One arithmetic operation that can be realized is matrix-vector multiplication.[20] As shown in Fig. 5(a), in order to perform $Ax = b$, the elements of A should be mapped linearly to the conductance values of PCM devices organized in a cross-bar configuration. The x values are encoded into the amplitudes or durations of read voltages applied along the rows. The positive and negative elements of A could be coded on separate devices together with a subtraction circuit, or negative vector elements could be applied as negative voltages. The resulting currents along the columns will be proportional to the result b. If inputs are encoded into durations, the result b is the total charge (e.g., current integrated over time). The property of the device that is used is the multi-level storage capability as well as the Kirchhoff circuit laws: Ohm's law and Kirchhoff's current law. The same cross-bar configuration can be used to perform a matrix-vector multiplication with the transpose of A. For this, the input voltage has to be applied to the column lines and the resulting current has to be measured along the rows. Mapping of the matrix elements to the conductance values of the resistive memory device can be achieved via iterative programming using the programming curve.[5] Figure 5(b) shows an experimental demonstration of a matrix-vector multiplication using real PCM devices fabricated in the 90 nm technology node. A is a 256×256 Gaussian matrix coded in a PCM chip and x is a 256-long Gaussian vector applied as voltages to the devices. It can be seen that the matrix-vector multiplication has a precision comparable to that of 4-bit fixed point arithmetic. This precision is mostly determined by the conductance fluctuations discussed in Sec. I.

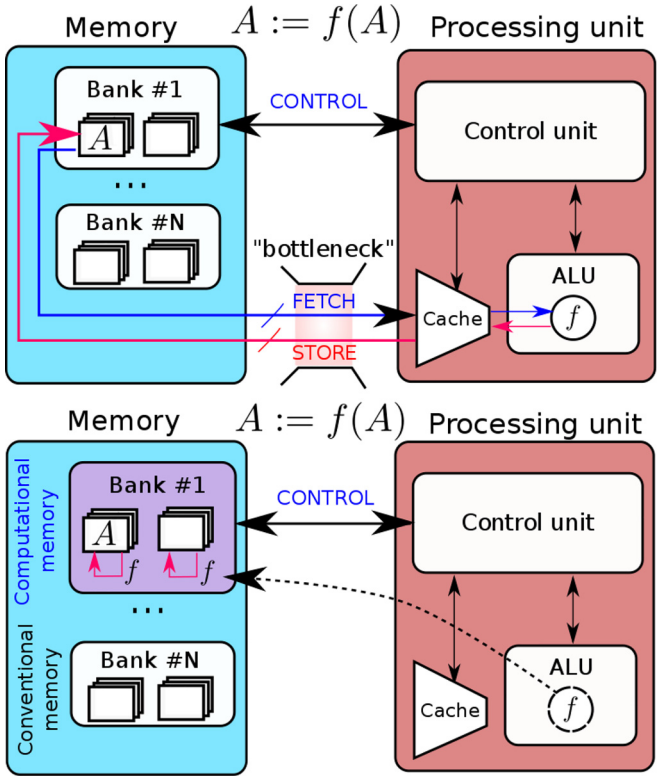

FIG. 4. "In-memory computing," computation is performed in place by exploiting the physical attributes of memory devices organized as a "computational memory" unit. For example, if data A is stored in a computational memory unit and if we would like to perform $f(A)$, then it is not required to bring A to the processing unit. This saves energy and time that would have to be spent in the case of conventional computing system and memory unit. Adapted from Ref. 19.

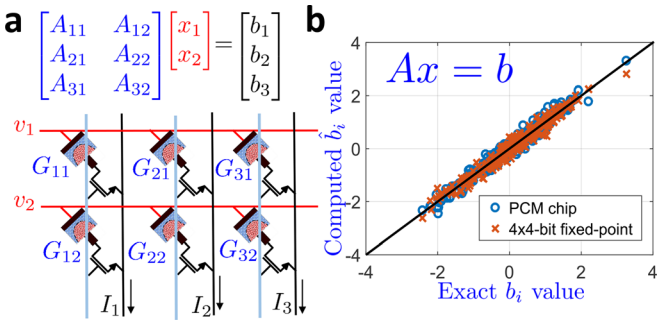

FIG. 5. (a) Matrix-vector multiplication can be performed with O(1) complexity by organizing PCM devices in a cross-bar configuration. (b) Experimental demonstration of the concept using PCM devices fabricated in the 90 nm technology node.

Compressed sensing and recovery is one of the applications that could benefit from a computational memory unit that performs matrix-vector multiplications.[21] The objective behind compressed sensing is to acquire a large signal at a sub-Nyquist sampling rate and 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 domains of medical imaging, security systems, and camera sensors.[22] 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 $\mathscr{M}$. The idea is to store this measurement matrix in the computational memory unit, with PCM devices organized in a cross-bar configuration [see Fig. 6(a)]. This allows us to perform the compression in O(1) time complexity. An approximate message passing algorithm (AMP) can be used to recover the original signal from the compressed measurements, using an iterative algorithm that involves several matrix-vector multiplications on the very same measurement matrix and its transpose. In this way, we can also use the same matrix that was coded in the computational memory unit for the reconstruction, reducing the reconstruction complexity from O(MN) to O(N). An experimental illustration of compressed sensing recovery in the context of image compression is shown in Fig. 6(b). 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 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 matrix-vector multiplications. 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.

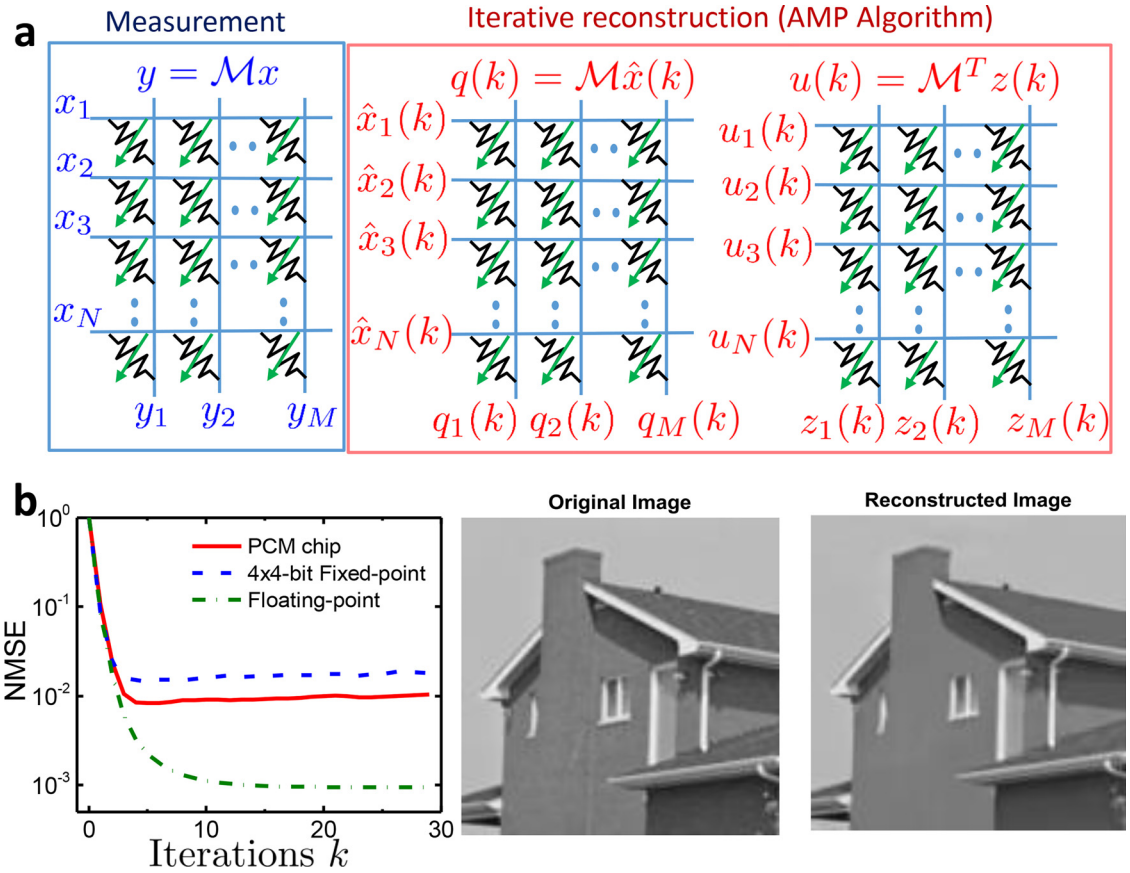

FIG. 6. (a) Compressed sensing involves one matrix-vector multiplication. Data recovery is performed via an iterative scheme, using several matrix-vector multiplications on the very same measurement matrix and its transpose. (b) An experimental illustration of compressed sensing recovery in the context of image compression is presented, showing 50% compression of a 128×128 pixel image. Adapted from Ref. 21. The normalized mean square error (NMSE) associated with the reconstructed signal is plotted against the number of iterations.

Another interesting demonstration of in-memory computing is that of unsupervised learning of temporal correlations between binary stochastic processes.[19] This problem arises in a variety of fields from finance to life sciences. Here, we exploit the accumulative behavior of the PCM devices. Each process is assigned to a PCM device as shown in Fig. 7(a). Whenever the process takes the value 1, a SET pulse is applied to the device. The amplitude of the SET pulse is chosen to be proportional to the instantaneous sum of all processes. With this procedure, it can be seen that the devices which are interfaced to the processes that are temporally correlated will go to a high conductance value. The simplicity of this approach belies the fact that a rather intricate operation of finding the sum of the elements of an uncentered covariance matrix is performed, using the accumulative behavior of the PCM devices. An experimental demonstration of the learning algorithm is presented involving a million pixels that are turning on and off, representing a million binary stochastic processes. Some of the pixels turn on and off with a weak correlation of $c = 0.01$, and the overall objective is to find them. Each pixel is assigned to a corresponding PCM device and the algorithm is executed as described earlier. It can be seen that after a certain period of time, the PCM devices associated with the correlated processes progress towards a high conductance value. This way, just by reading back the conductance values, we can decipher which of the binary random processes are temporally correlated [Fig. 7(b)]. The computation is massively parallel, with the final result of the computation imprinted onto the PCM devices. The reduction in computational time complexity is from $O(N)$ to $O(k \log(N))$, where k is a small constant and N is the number of data streams. A detailed system-level comparative study with respect to state-of-the-art computing hardware was also performed.[19] Various implementations were compiled and executed on an IBM Power System S822LC system with 2 Power8 central processing units (CPUs) (each comprising 10 cores) and 4 Nvidia Tesla P100 graphical processing units (GPUs) attached using the NVLink Interface. A multi-threaded implementation was designed that can leverage the massive parallelism offered by the GPUs, as well as a scale out implementation that runs across several GPUs. For the PCM, a write latency of 100 ns and a programming energy of 1.5 pJ were assumed for each SET operation. It was shown that using such a computational memory module, it is possible to accelerate the task of correlation detection by a factor of 200 relative to an implementation that uses 4 state-of-the-art GPU devices. Moreover, power profiling of the GPU implementation indicates that the improvement in energy consumption is over two orders of magnitude.

The compressed sensing recovery and unsupervised learning of temporal patterns are two applications that

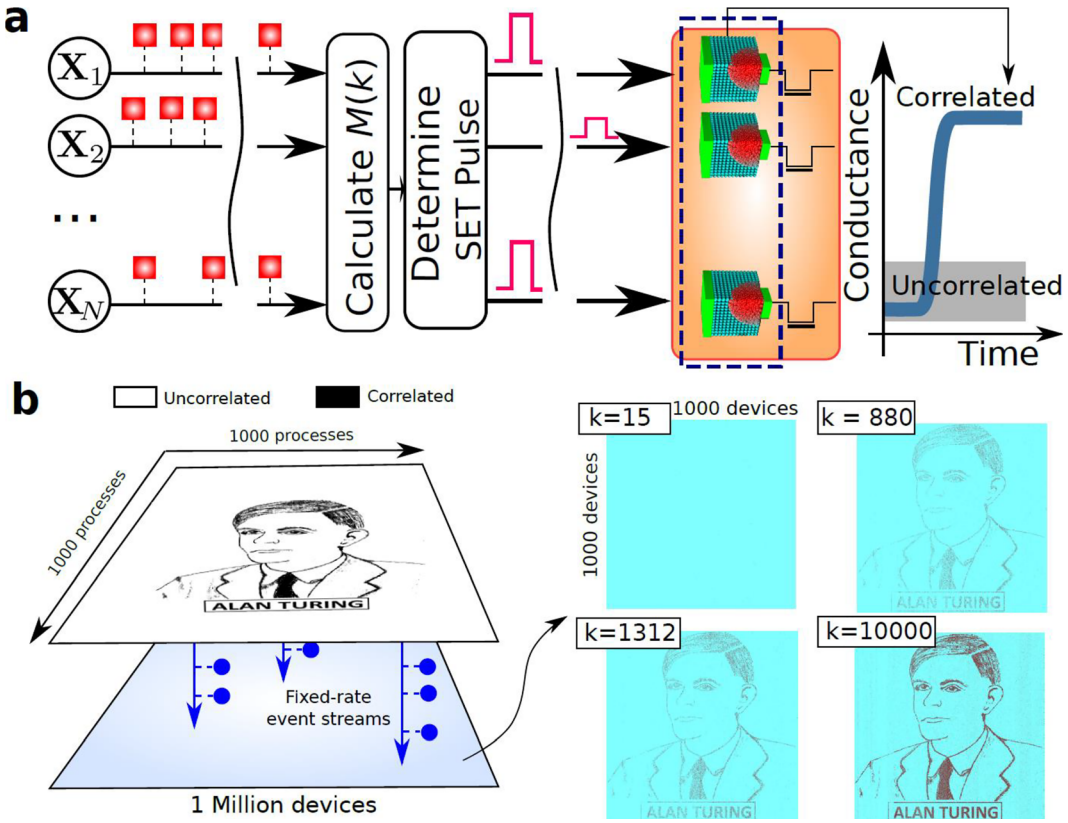

FIG. 7. (a) By interfacing binary random processes to PCM devices, it is possible to learn the temporal correlations between the processes in an unsupervised manner. (b) Experimental demonstration of the concept using a million PCM devices.[19] The result of the computation is imprinted as conductance values in the memory array.

clearly demonstrate the potential of PCM-based computational memory in tackling certain data-centric computational tasks. The former exploits the multi-level storage capability, whereas the latter mostly relies on the accumulative behavior. However, one key challenge associated with computational memory is the lack of high precision. Even though approximate solutions are sufficient for many computational tasks in the domain of AI, there are some applications that require that the solutions are obtained with arbitrarily high accuracy. Fortunately, many such computational tasks can be formulated as a sequence of two distinct parts. In the first part, an approximate solution is obtained; in the second part, the resulting error in the overall objective is calculated accurately. Then, based on this, the approximate solution is refined by repeating the first part. Step I typically has a high computational load, whereas Step II has a light computational load. This forms the foundation for the concept of mixed-precision in-memory computing: the use of a computational memory unit in conjunction with a high-precision von Neumann machine.[23] The low-precision computational memory unit can be used to obtain an approximate solution as discussed earlier. The high-precision von Neumann machine can be used to calculate the error precisely. The bulk of the computation is still realized in computational memory, and hence we still achieve significant areal/power/speed improvements while addressing the key challenge of imprecision associated with computational memory).

A practical application of mixed-precision in-memory computing is that of solving systems of linear equations (if $Ax = b$, find x). As shown in Fig. 8(a), an initial solution is chosen as a starting point and is then iteratively updated based on a low-precision error-correction term, z. This error-correction term is computed by solving $Az = r$ with an inexact inner solver, using the residual $r = b - Ax$ calculated with high precision. The matrix multiplications in the inner solver are performed inexactly using computational memory. The algorithm runs until the norm of the residual falls below a desired pre-defined tolerance, *tol*. An experimental demonstration of this concept using model covariance matrices is shown in Fig. 8(b). The model covariance matrices exhibit a decaying behavior that simulates the decreasing correlation of features away from the main diagonal. The matrix multiplications in the inner solver are performed using PCM devices. The norm of the error between the estimated solution and the actual solution is plotted against the number of iterative refinements. It can be seen that for all matrix dimensions, the accuracy is not limited by the precision of the computational memory unit. Several system-level measurements using Power 8 CPUs and P100 GPUs serving as the high-precision processing unit showed that up to 6.8× improvements in time/energy to solution can be achieved for large matrices. Moreover, this gain can increase to more than one order of magnitude for more accurate computational memory units.

Computational memory can be viewed as a natural extension of conventional memory units, either in a system-on-a-chip (SoC) or as a stand-alone module. The objective of such a unit is to perform certain relatively generic

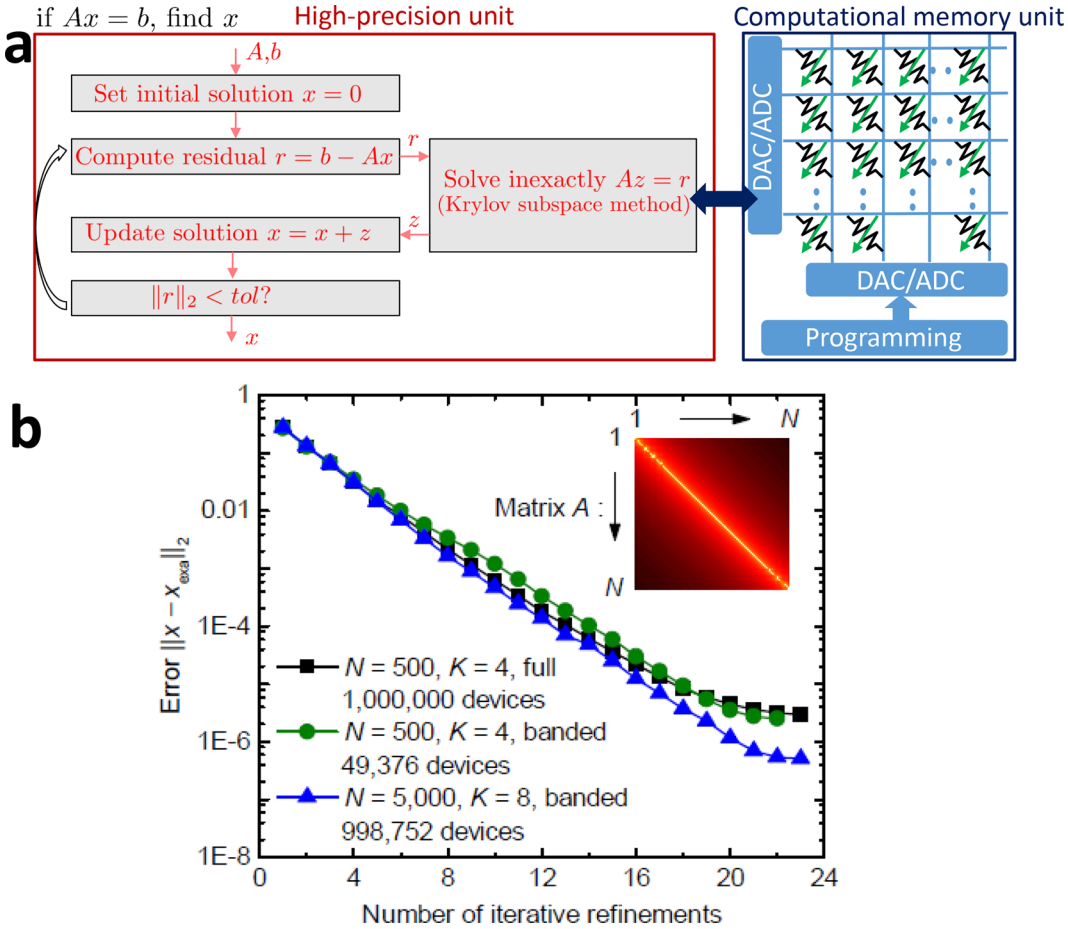

FIG. 8. (a) Mixed-precision in-memory computing architecture to solve systems of linear equations. (b) Experimental demonstration of the concept using model covariance matrices.[23]

computational primitives in place with remarkably high efficiency, but in conjunction with the other components of a computing system. In Secs. III and IV, we discuss non-von Neumann co-processors realized based on an underlying neural network framework.

III. DEEP LEARNING CO-PROCESSORS

Recently, deep artificial neural networks have shown remarkable human-like performance in tasks such as image processing and voice recognition. Deep neural networks are loosely inspired by biological neural networks. Parallel processing units called neurons are interconnected by plastic synapses. By tuning the weights of these interconnections, these networks are able to solve certain problems remarkably well. The training of these networks is based on a global supervised learning algorithm typically referred to as back-propagation. 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 back-propagated. Both steps involve sequences of matrix-vector multiplications. Subsequently, the synaptic weights are updated to reduce the error. Because of the need to repeatedly show very large datasets to very large neural networks, this brute force optimization approach can take multiple days or weeks to train state-of-the-art networks on von Neumann machines.

The mixed-precision in-memory computing concept can be extended to the problem of training deep neural networks, where a computational memory unit is used to perform the forward and backward passes, while the weight changes are accumulated in high precision.[26] However, one could also envisage a co-processor comprising multiple cross-bar arrays of PCM devices and other analog communication links and peripheral circuitry to accelerate all steps of deep learning.[24] This could be viewed as a second level of brain-inspired computing using PCM devices. The essential idea is to represent the synaptic weights associated with each layer in terms of the conductance values of PCM devices organized in a cross-bar configuration. There will be multiple such cross-bar arrays corresponding to the multiple layers of the neural network. Such a co-processor also comprises the necessary peripheral circuitry to implement the neuronal activation functions and communication between the cross-bar arrays.

This deep learning co-processor concept is best illustrated with the help of an example. Let us consider the problem of training a neural network to classify handwritten digits based on the MNIST dataset. As shown in Fig. 9, a network with two fully connected synaptic layers is chosen.

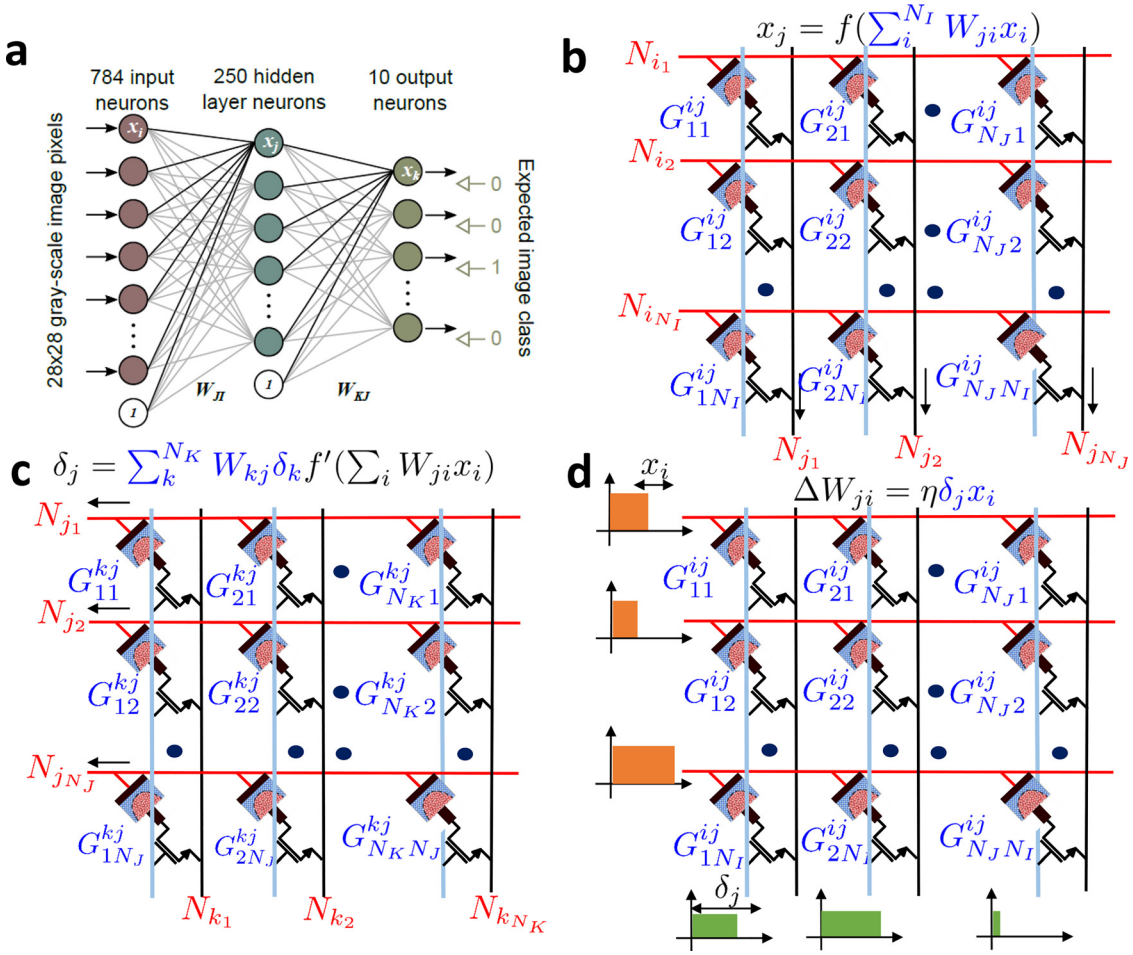

FIG. 9. (a) A prototypical two-layer neural network to classify handwritten digits based on the MNIST dataset. (b) The matrix-vector multiplications associated with the forward pass can be implemented with O(1) complexity. (c) The backward pass involves a multiplication with the transpose of the matrix representing the synaptic weights, which can be realized with O(1) complexity. (d) The synaptic weight update can also be achieved in place in O(1) time complexity.

The number of neurons in the input, hidden and output layers is 784, 250, and 10, respectively. The synaptic weights associated with the two layers are stored in two different cross-bar arrays. The matrix-vector multiplications associated with the forward and backward passes can be implemented efficiently in O(1) complexity as described earlier. It is also possible to induce the synaptic weight changes in O(1) complexity by exploiting the cross-bar topology. Figure 10(a) shows the training and test accuracies corresponding to a mixed hardware/software demonstration of this concept.

The test accuracy of 82.9% demonstrated here is not particularly high, which can be attributed to the non-idealities of the PCM devices discussed in Sec. I. The most critical device requirement for backpropagation training is the need for symmetric weight update. If the algorithm increases (decreases) some given weight within the neural network, and then later requests a counteracting decrease (increase) of that same weight, those two separate conductance programming events must cancel on average.[24,27] Unfortunately, the nonlinearity associated with the accumulative behavior creates a consistent bias.[24] In a recent experiment, compact "3T1C" circuit structures that combine 3 transistors with 1 capacitor greatly increased the linearity and the granularity of the weight update, allowing PCM devices to be used for the non-volatile storage of weight data transferred from the 3T1C structures.[25] Since this weight-transfer process is performed via iterative programming, the programming accuracy of weights no longer depends on PCM conductance nonlinearity or device-to-device variability, although it is still affected by the inherent stochasticity and conductance fluctuations. In spite of using the same PCM devices used in the 2014 experiment, the classification accuracy was shown to increase the accuracy of the mixed-hardware-software experiment to software-equivalent levels [see Figs. 10(b) and 10(c)].

A proposed chip architecture for such a co-processor is shown in Fig. 11. The architecture is composed of a large number of identical array-blocks connected by a flexible routing network. Each array-block here represents a large PCM device array. A flexible routing network has three tasks: (1) to convey chip inputs (such as example data, example labels, and weight overrides) from the edge of the chip to the device arrays, (2) to carry chip outputs (such as inferred classifications and updated weights) from the arrays to the edge of the chip, and (3) to interconnect various arrays in order to implement multi-layer neural networks. Each array has input neurons (here shown on the "West" side of each array) and output neurons ("South" side), connected with a

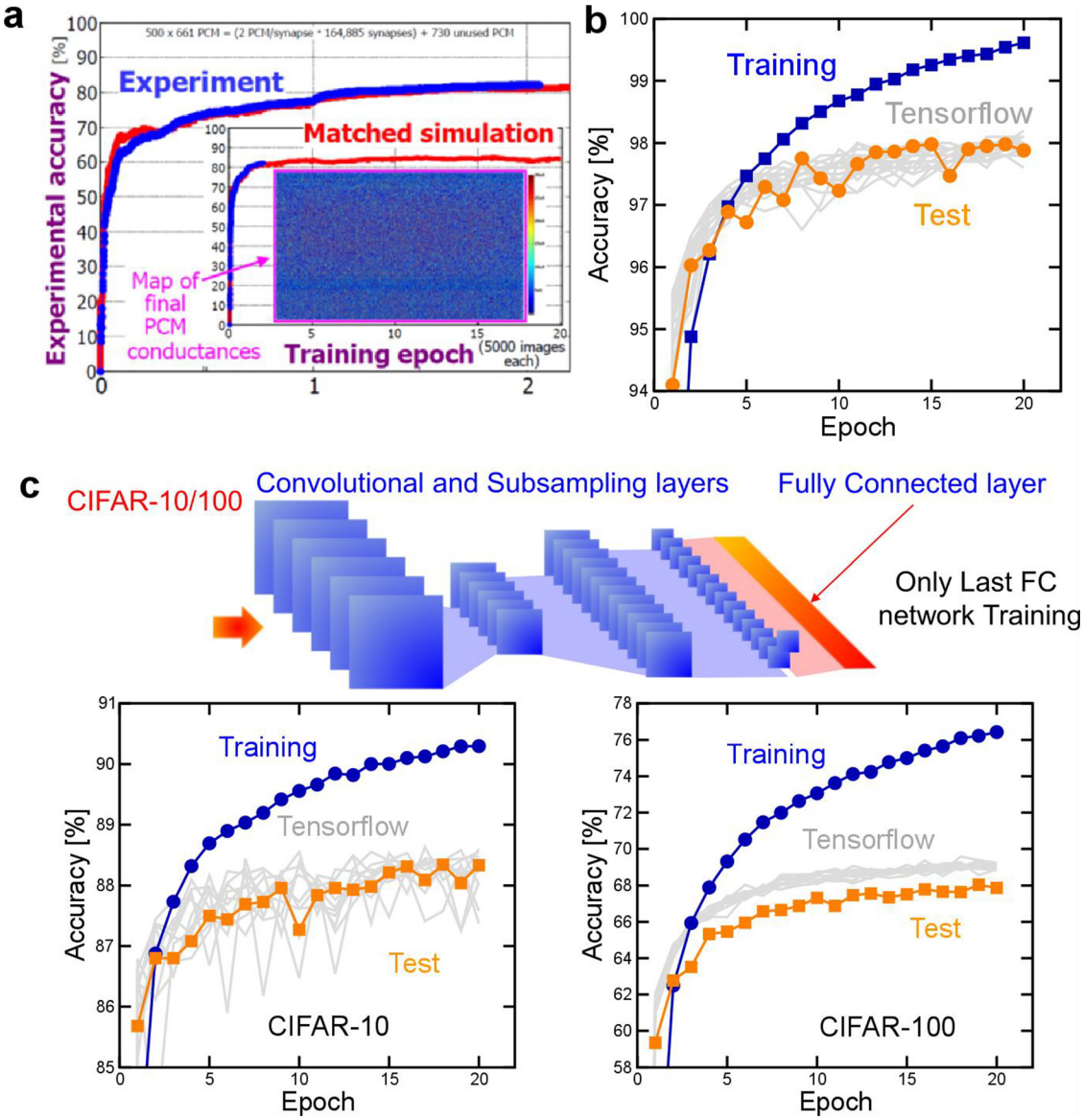

FIG. 10. (a) A mixed hardware/software demonstration of the concept of training deep neural networks using PCM devices. The training and test accuracies as a function of the number of training epochs are shown.[24] (b) and (c) The classification accuracy is shown to increase the accuracy of the mixed-hardware-software experiment to software-equivalent levels with the use of a "3T1C" circuit structure in conjunction with the PCM devices,[25] both for (b) MNIST handwritten digits and for (c) transfer learning experiments involving CIFAR-10 and CIFAR-100 datasets.

dense grid of synaptic connections. Peripheral circuitry is divided into circuitry assigned to individual rows and columns, circuitry shared between a number of neighboring rows and columns, and support circuitry. Power estimations for device arrays and the requisite analog peripheral circuitry project power per DNN training example as low as 44 mW, for a computational energy efficiency of 28 065 Giga-Operations per second per Watt (GOP/s/W) and a throughput-per-unit-area of 3.6 Tera-Operations per second per square millimeter (TOP/s/mm^2) $\approx$ 280$\times$ and 100$\times$ better than the most recent GPU models, respectively.[25]

The next steps will be to design, implement and refine these analog techniques on prototype PCM-based hardware accelerators and to demonstrate software-equivalent training accuracies on larger networks. Since the most efficient mapping offered by crossbar arrays of PCM or other analog memory devices is to large, fully connected neural network layers, one suitable class of networks is recurrently connected Long Short Term Memory (LSTM)[29] and Gated Recurrent Unit (GRU)[30] networks behind recent advances in machine translation, captioning and text analytics.

IV. SPIKING NEURAL NETWORKS

Despite our ability to train deep neural networks with brute-force optimization, the computational principles of neural networks remain poorly understood. Hence, significant research is aimed at unravelling the principles of computation in large biological neural networks and, in particular, biologically plausible spiking neural networks (SNNs). In biological neurons, a thin lipid-bilayer membrane separates the electrical charge inside the cell from that outside it. This allows an equilibrium membrane potential to be maintained in conjunction with several electrochemical

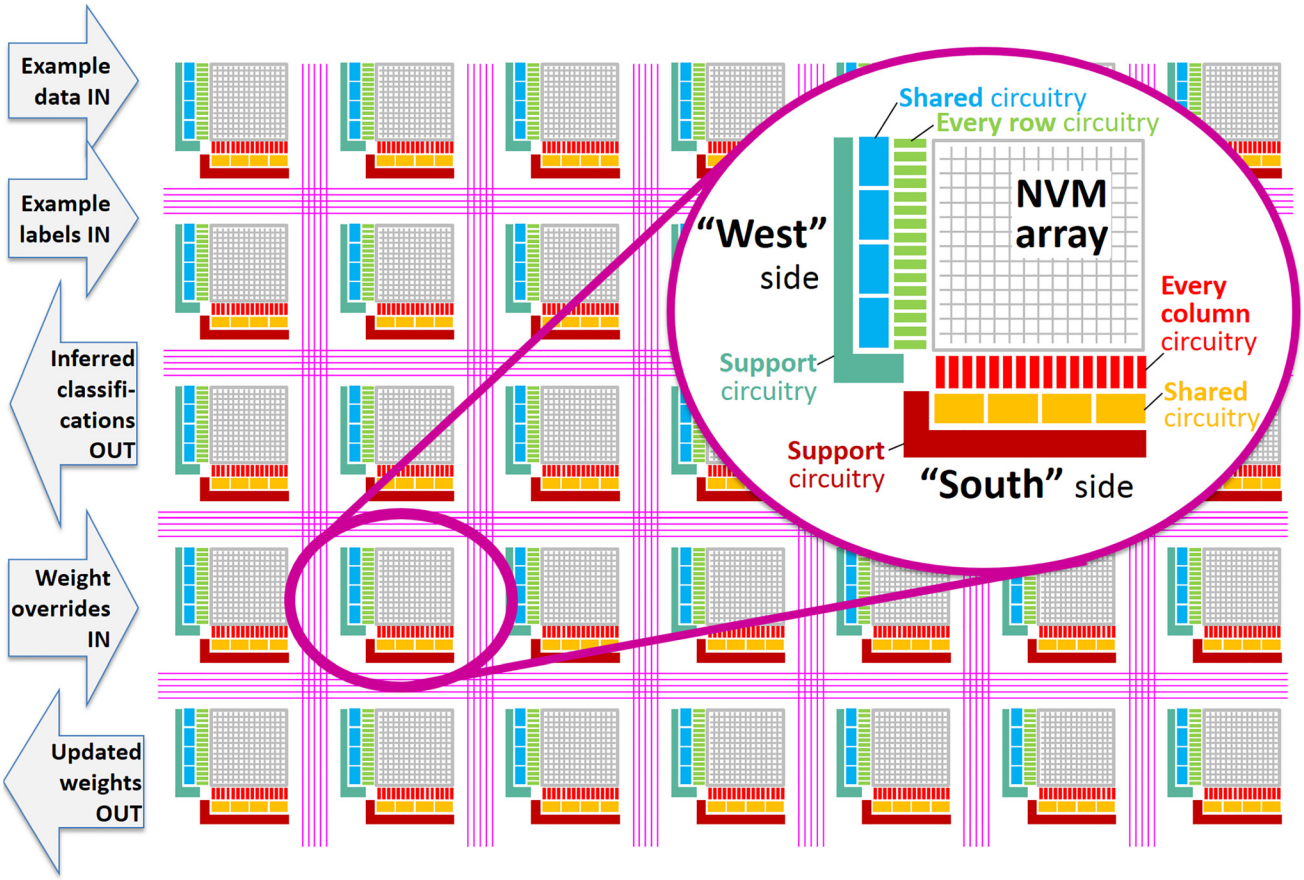

FIG. 11. A proposed chip architecture for a co-processor for deep learning based on PCM arrays.[28]

mechanisms. However, this membrane potential could be changed by the excitatory and inhibitory input signals through the dendrites of the neuron. Upon sufficient excitation, an action potential is generated, referred to as neuronal firing or spike generation. The neurons pass this firing information to other neurons through synapses. There are two key attributes associated with these synapses: synaptic efficacy and synaptic plasticity. Let us consider two such neurons connected to each other via synaptic connections [see Fig. 12(a)]. Synaptic efficacy refers to the generation of a synaptic output based on the incoming neuronal activation and is indicative of the strength of the connection between the two neurons denoted by the synaptic weight. For example, in response to a pre-synaptic neuronal spike, a postsynaptic potential is generated and then serves as an input to a dendrite of the post-synaptic neuron. Synaptic plasticity, in contrast, is the ability of the synapse to change its weight, typically in response to the pre- and post-synaptic neuronal spike activity. A well-known plasticity mechanism is spike-time-dependent plasticity (STDP), where synaptic weights are changed depending on the relative timing between the spike activity of the input (pre-synaptic) and output (post-synaptic) neurons [see Fig. 12(b)].

Highly specialized computational platforms are required to realize these neuronal and synaptic dynamics and their interconnections in an efficient manner. Most of the efforts in building such computational substrates to date are based on digital and analog CMOS circuitry.[32–37] In 2014, IBM presented a million spiking-neuron chip with a scalable communication network and an interface.[33] The chip, TrueNorth, has 5.4×10^9 transistors, 4096 neuro-synaptic cores and 256×10^6 configurable synapses. However, this chip does not perform *in-situ* learning. An alternate approach is to exploit the subthreshold MOSFET characteristics to directly emulate the biophysics of neural systems.[35] In particular, this can be achieved by using field-effect transistors (FETs) operated in the analog weak-inversion or "subthreshold" domain. These naturally exhibit exponential relationships in their transfer functions, similar to the exponential dependencies observed in the conductance of sodium and potassium channels of biological neurons.

PCM devices could also play a key role in the space of specialized computing substrates for SNNs. This can be viewed as a third level of brain-inspired computing using these devices. A particularly interesting application is in the emulation of neuronal and synaptic dynamics. The essential idea of phase-change neurons is to realize the neuronal dynamics using the accumulative behavior resulting from the crystallization dynamics.[38] The internal state of the neuron is represented in terms of the phase configuration of the PCM device (see Fig. 13). By translating the neuronal input signals into appropriate electrical signals, it is possible to tune the firing frequency in a highly controllable manner proportional to the strength of the input signals.

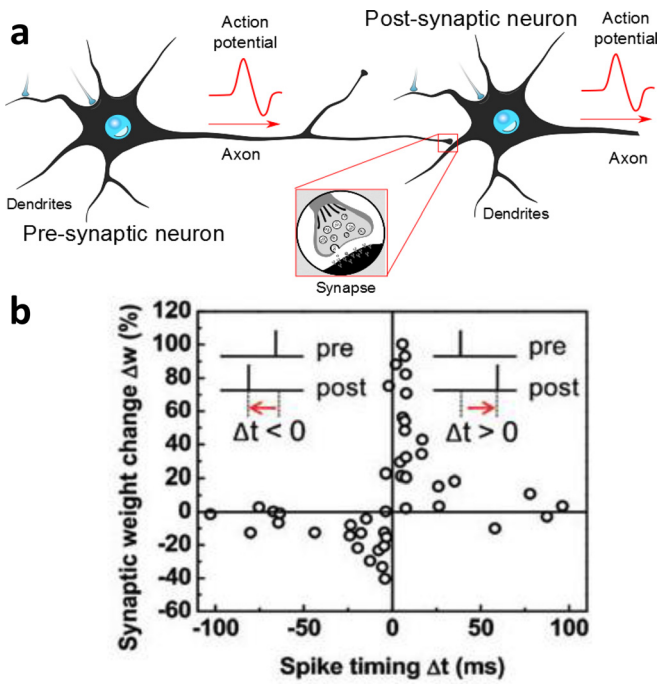

FIG. 12. (a) Schematic illustration of a synaptic connection and the corresponding pre- and post-synaptic neurons. The synaptic connection strengthens or weakens based on the spike activity of these neurons; a process referred to as synaptic plasticity. (b) A well-known plasticity mechanism is spike-time-dependent plasticity (STDP), leading to weight changes that depend on the relative timing between the pre- and post-synaptic neuronal spike activities. Adapted from Ref. 31.

In addition to the deterministic neuronal dynamics, stochastic neuronal dynamics also play a key role in signal encoding and transmission in biological neural networks. The use of neuronal populations to represent and transmit sensory and motor signals is one prominent example. The stochastic neuronal dynamics is attributed to a number of complex phenomena, such as ionic conductance noise, chaotic motion of charge carriers due to thermal noise, interneuron morphologic variabilities, and other background noise.[39] It has been shown that emulating this stochastic firing behavior within artificial neurons could enable intriguing functionality.[40] Tuma *et al.* showed that neuronal realizations using PCM devices exhibit significant interneuronal as well as intra-neuronal randomness, thus mimicking this stochastic neuronal behavior at the device level. The intra-neuronal stochasticity arises from the randomness associated with the accumulative behavior as discussed in Sec. I. This causes multiple integrate and-fire cycles in a single phase-change neuron to generate a distribution of interspike intervals, thus enabling population-based computation. Fast signals were demonstrated to be accurately represented by overall neuron population, despite the rather slow firing rate of the individual neurons.[38]

The ability to alter the conductance levels in a controllable way makes PCM devices particularly well-suited for synaptic realizations. The synaptic weights can be represented in terms of the conductance states of PCM devices. Synaptic efficacy can be emulated by biasing the devices with a suitable voltage signal initiated by a pre-synaptic neuronal spike. The resulting read current could represent the post-synaptic potential, which in turn can be propagated to the post-synaptic neurons. It is also possible to emulate synaptic plasticity in a very elegant manner. For example, in one implementation of STDP, the pre-synaptic neuronal spike initiates a sequence of pulses with varying amplitude and the post-synaptic neuronal spike initiates a single pulse with opposite polarity [see Fig. 14(a)]. The pulse amplitudes are chosen such that the PCM devices are programmed when the pulse corresponding to the post-synaptic neuronal spike overlaps with one of the pulses corresponding to the pre-synaptic neuronal spike and depending on the relative time difference between the spikes, the PCM device conductance is increased or decreased.[42] With the access transistor playing a more active role, the STDP rule can be implemented more efficiently [see Fig. 14(b)]. In this realization, the pre-synaptic neuronal spike initiates a voltage pulse applied to the gate of the transistor and the post-synaptic neuronal spike initiates a pulse applied to the top electrode of the PCM device. The shape of the pulse waveform is chosen such that it implements the desired STDP rule. The FET only permits programming (and the associated energy consumption) during the brief overlap between the two signals.[43] However, a significant drawback of a single PCM-based synapse is that it is not possible to progressively depress as was discussed in Sec. I. The solution is to realize a single synapse using two PCM devices organized in a differential configuration.[44] Here, one PCM device realizes the long-term synaptic potentiation (LTP), while the other helps to realize the long-term synaptic depression (LTD) [see Fig. 14(c)]. Both LTP and LTD devices receive potentiating pulses and the currents flowing through the LTD PCM is subtracted from that flowing through the LTP PCM in the post-synaptic neuron. When the devices are saturated or when they reach their minimum resistance value, they have to be periodically reset and reprogrammed. More recently, it was shown that the two key synaptic attributes of efficacy and plasticity can be efficiently realized using a unit comprising 1 PCM device and 2 transistors (see Fig. 15). This is achieved by turning ON the appropriate transistor as well as the application of suitable electrical pulses. A neuromorphic core comprising 64 000 such synaptic elements was also fabricated.[41] A top-level schematic is shown in Fig. 15(b).

A single PCM neuron can be interfaced with several PCM synapses to realize simple all-PCM neural networks that can detect spatio-temporal patterns in an unsupervised manner.[38,46–48] The input is fed as a sequence of spikes and a local STDP learning rule is implemented at the synaptic level. The single neuron based neural networks can be extended to multiple neurons. With an additional winner-take-all (WTA) mechanism, pattern classification tasks can be performed in an unsupervised manner.[49] We present an example where such a network is used to classify handwritten digit database.[45] The task is identical to that described in Sec. III, but in this case, the classification task is performed in an unsupervised manner. A local learning rule is employed as opposed to the global backpropagation algorithm. The network consists of a single layer with all-to-all synaptic connections as shown in Fig. 16(a). There are 50 output neurons, n_j, implementing the leaky integrate-and-fire

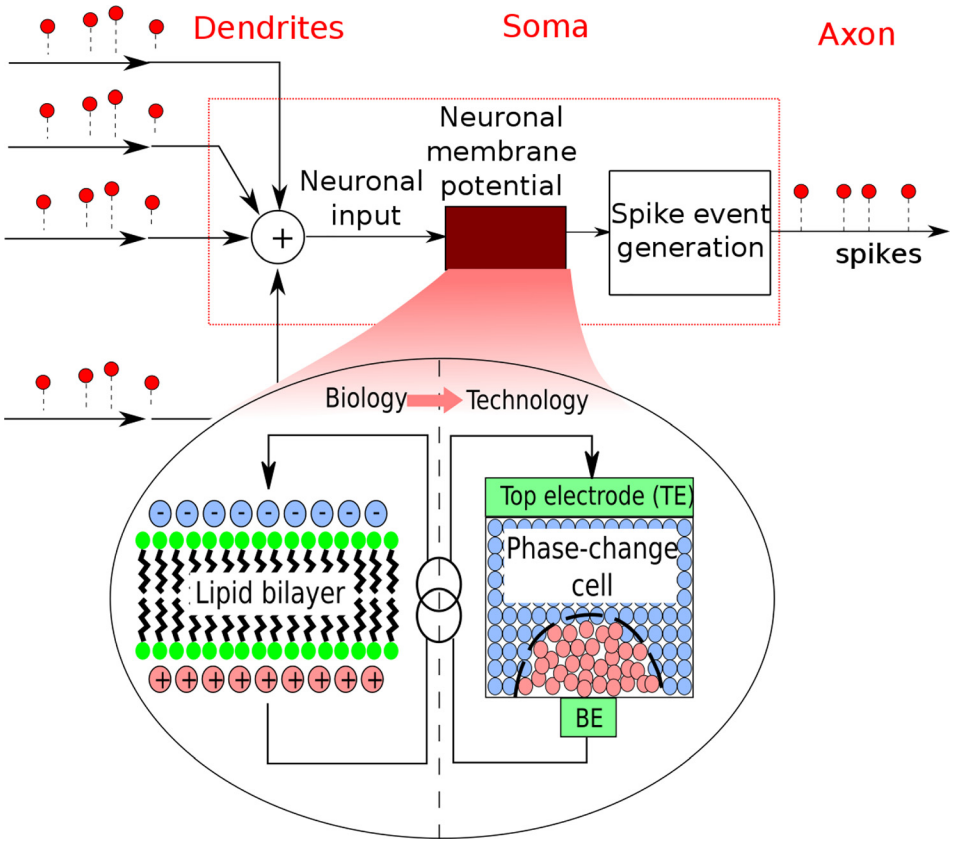

FIG. 13. PCM devices can be used to emulate the neuronal dynamics.[38] The phase configuration within the PCM device is used to represent the internal state of the neuron.

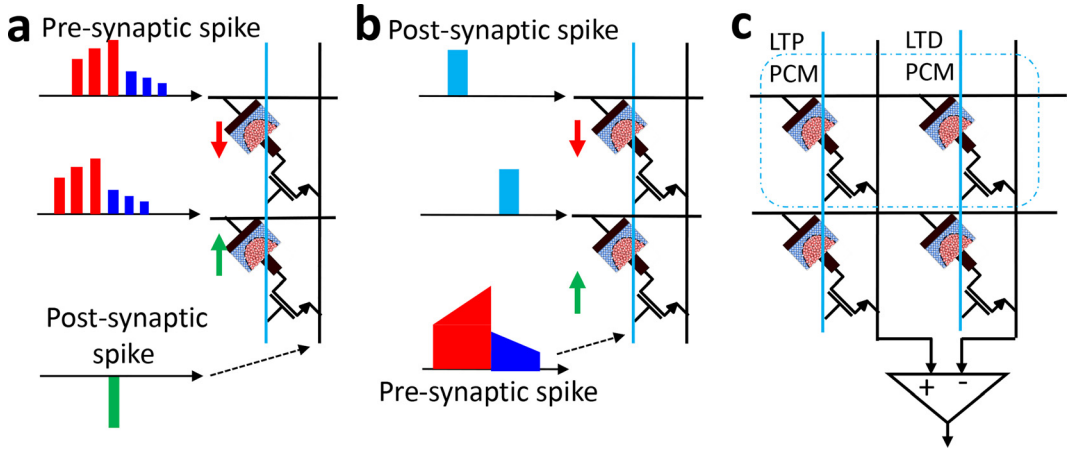

FIG. 14. (a) Two synaptic elements, each comprising a PCM device and a transistor. The transistor just serves as a switch in this configuration. To implement STDP, the pre-synaptic neuronal spike initiates a sequence of pulses with varying amplitude and the post-synaptic neuronal spike initiates a single pulse with opposite polarity. (b) Two synaptic elements, each comprising a PCM device and a transistor. In this configuration, the transistor plays an active role in the realization of synaptic plasticity. The pre-synaptic neuronal spike initiates a voltage pulse applied to the gate of the transistor and the post-synaptic neuronal spike initiates a pulse applied to the top electrode of the PCM device. (c) A synaptic element comprising two PCM devices organized in a differential configuration.

model. These neurons are interfaced to the synaptic elements that receive as input patterns consisting of 28×28 pixel grayscale images that are presented to the network using a rate-encoding scheme. Specifically, the pixel intensity is linearly mapped to a frequency which serves as the mean frequency of a random Poisson process to generate the input spikes, x_i. There are two steps associated with the learning task. In the first step, the network clusters the inputs in an unsupervised way with each neuron responsible for one cluster. In the second step, every cluster is assigned to one of the digit classes using the appropriate labels. In the first step, a winner-take-all (WTA) mechanism is employed to introduce competition among the output neurons. The WTA scheme selects one winning neuron among all the neurons that cross the firing threshold based on the difference between the respective membrane potential and the firing threshold. Moreover, the threshold voltages are adapted to their respective stimuli using homeostasis to ensure that all the neurons participate in the learning process. A modified STDP algorithm is used for the learning. Two time windows defined as

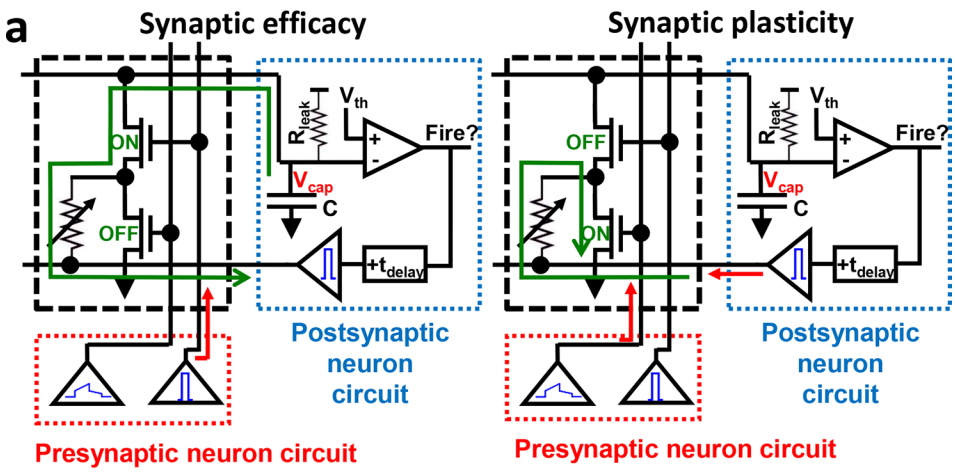

FIG. 15. (a) Synaptic efficacy and plasticity can be realized very efficiently using a synaptic element comprising 1 PCM device and two transistors. (b) A neuromorphic core comprising 64 000 such synaptic elements.[41]

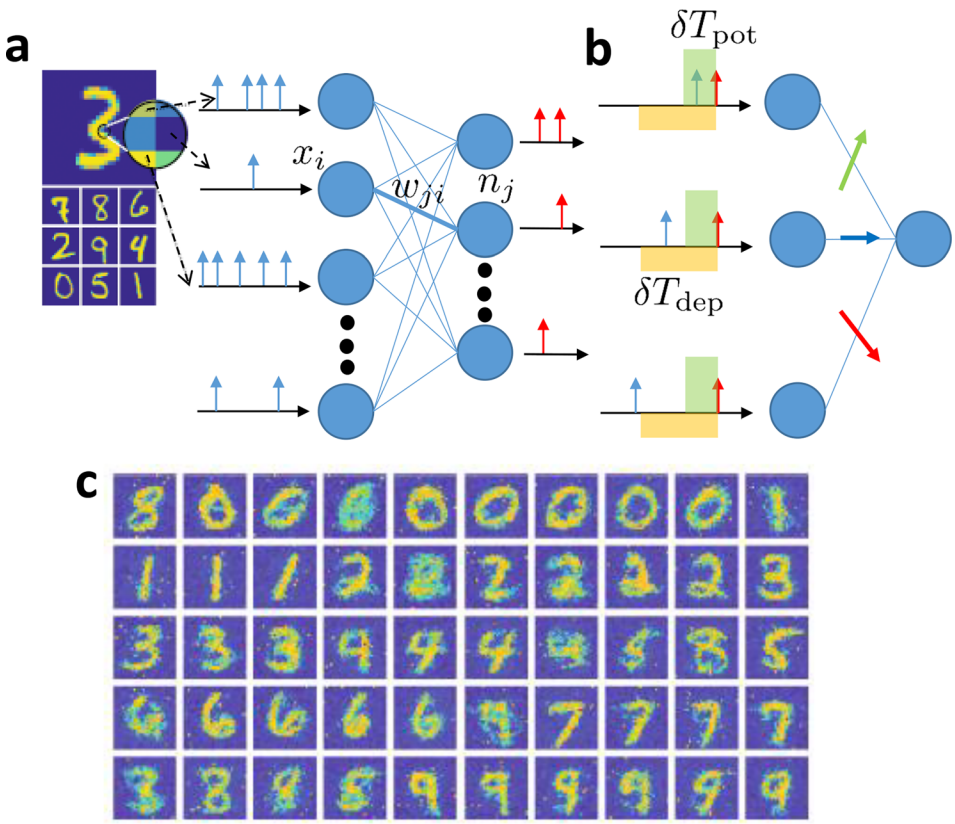

FIG. 16. (a) A spiking neural network architecture for unsupervised learning of handwritten digits. (b) Schematic illustration of the modified STDP rule. (c) The synaptic weight map corresponding to the 50 output neurons as stored in the PCM-based synapses through the unsupervised learning scheme.[45]

δT_{pot} and δT_{dep} are shown in Fig. 16(b). When an output neuron n_j spikes at a time instant, t_j, the corresponding synaptic weights are modified depending on the time, t_i of their last input spike. If $t_j - t_i < \delta T_{pot}$, the synapse w_{ji} gets potentiated. In the case, where $t_j - t_i > \delta T_{dep}$, the synapse gets depressed. In all other cases, the synaptic weight remains unchanged. This network was implemented experimentally where PCM devices are used to implement the synapses (2 PCM devices in a differential configuration to denote one synapse), while the learning rule and the neurons were emulated in software. The synaptic weight map corresponding to the 50 output neurons are shown in Fig. 16(c). This experiment achieved a test accuracy of 68.14%, which is quite remarkable given that this is an unsupervised learning task and real PCM devices were used to represent the synaptic weights.

It is widely believed that because of the added temporal dimension, SNNs should be computationally more powerful.[50,51] The asynchronous nature of computation also makes them particularly attractive for temporarily sparse data. However, a killer application that transcends conventional deep learning as well as a robust scalable global training algorithm that can harness the local SNN learning rules are still lacking. Hence, algorithmic exploration has to go hand-in-hand with advances in the hardware front. For example, there are recent results that show that one could learn efficiently from multi-timescale data with the addition of a short term plasticity rule to STDP.[52,53]

V. DISCUSSION AND OUTLOOK

The brain-inspired computing schemes described so far are expected to reduce the time, energy and area required to arrive at a solution for a number of AI-related applications. System-level studies show that even with today's PCM technology, we can achieve significantly higher performance compared to conventional approaches.[23] There are also strong indications that this performance improvement will be substantially higher with future generations of PCM devices. Phase-change materials are known to undergo reversible phase transition down to nanoscale dimensions with substantially lower power.[54] Reducing the programming current will also help reduce the size of the access device in the case of 1T1R configurations. However, circuit-level aspects such as the voltage drop across the long wires connecting the devices could still limit the achievable areal density. There are also phase-change materials that can undergo phase transition on the order of nanoseconds.[55] This could significantly increase the efficiency and the performance of PCM-based computing systems. Moreover, the retention time, which is a key requirement for the traditional memory application is not so critical for several computing applications and this could enable the exploration of new material classes. For example, it was recently shown that single elemental antimony could be melt-quenched to form a stable amorphous state at room temperature.[56]

However, there are also numerous roadblocks associated with using PCM devices for computational purposes. One key challenge applicable to almost all the applications in brain-inspired computing is the variation in conductance values arising from $1/f$ noise as well as structural relaxation of the melt-quenched amorphous phase. There are also temperature-induced conductance variations. One very promising research avenue towards addressing this challenge is that of projected phase-change memory.[57,58] These devices provide a shunt path for read current to bypass the amorphous phase-change material. Another challenge is the limited endurance of PCM devices (the number of times the PCM devices can be SET and RESET), which is relatively high (approx. 10^9–10^{12}),[59] but may not be adequate for certain computational applications. The non-linearity and stochasticity associated with the accumulative behavior are key challenges, in particular, for applications involving *in-situ* learning. Multi-PCM architectures could partially address these challenges.[60] However, more research in terms of device geometries and randomness associated with crystal growth is required.

We conclude with an outlook towards the adoption of PCM-based computing systems in future AI hardware. Current research on AI hardware is mostly centered around conventional von Neumann architecture. The overarching objective is to minimize the time and distance to memory access so that the von Neumann bottleneck is alleviated to a large extent. One approach is to improve the memory/storage hierarchy by introducing new types of memory such as storage class memory.[61,62] Near-memory computing is another approach where CMOS processing units are placed in close proximity to the memory unit.[63] There is also significant research activity in the space of custom ASICs (highly power/area optimized) for various AI applications, in particular, deep learning.[64] Unlike all these research efforts, the computational approaches presented in this tutorial are distinctly non-von Neumann in nature. By augmenting conventional computing systems, these systems could help achieve orders of magnitude improvement in performance and efficiency. In summary, we believe that we will see two stages of innovations that take us from the near term, where the AI accelerators are built with conventional CMOS, towards a period of innovation involving the computational approaches presented in this article.

ACKNOWLEDGMENTS

We acknowledge the contributions of our colleagues, in particular, Angeliki Pantazi, Giovanni Cherubini, Stanislaw Wozniak, Timoleon Moraitis, Irem Boybat, S. R. Nandakumar, Wanki Kim, Pritish Narayanan, Robert M. Shelby, Stefano Ambrogio, and Hsinyu Tsai. A.S. would like to acknowledge funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant Agreement No. 682675).

[1]G. W. Burr *et al.*, IEEE J. Emerging Sel. Top. Circuits Syst. **6**, 146 (2016).
[2]S. Raoux, D. Ielmini, M. Wuttig, and I. Karpov, MRS Bull. **37**, 118 (2012).
[3]N. Papandreou, A. Pantazi, A. Sebastian, M. Breitwisch, C. Lam, H. Pozidis, and E. Eleftheriou, in *International Conference on Electronics, Circuits, and Systems (ICECS)* (IEEE, 2010), pp. 1017–1020.
[4]N. Papandreou, A. Pantazi, A. Sebastian, E. Eleftheriou, M. Breitwisch, C. Lam, and H. Pozidis, Solid-State Electron. **54**, 991 (2010).

[5]N. Papandreou, H. Pozidis, A. Pantazi, A. Sebastian, M. Breitwisch, C. Lam, and E. Eleftheriou, in *International Symposium on Circuits and Systems (ISCAS)* (IEEE, 2011), pp. 329–332.
[6]D. Ielmini, Phys. Rev. B **78**, 035308 (2008).
[7]M. Le Gallo, A. Athmanathan, D. Krebs, and A. Sebastian, J. Appl. Phys. **119**, 025704 (2016).
[8]M. Nardone, V. Kozub, I. Karpov, and V. Karpov, Phys. Rev. B **79**, 165206 (2009).
[9]I. V. Karpov *et al.*, J. Appl. Phys. **102**, 124503 (2007).
[10]M. Le Gallo, D. Krebs, F. Zipoli, M. Salinga, and A. Sebastian, Adv. Electron. Mater. (published online 2018).
[11]M. Le Gallo, M. Kaes, A. Sebastian, and D. Krebs, New J. Phys. **17**, 093035 (2015).
[12]A. Sebastian, M. Le Gallo, and D. Krebs, Nat. Commun. **5**, 4314 (2014).
[13]M. Le Gallo, T. Tuma, F. Zipoli, A. Sebastian, and E. Eleftheriou, in *European Solid-State Device Research Conference (ESSDERC)* (IEEE, 2016), pp. 373–376.
[14]I. Boybat, M. L. Gallo, T. Moraitis, Y. Leblebici, A. Sebastian, and E. Eleftheriou, in *2017 13th Conference on Ph. D. Research in Microelectronics and Electronics (PRIME)* (IEEE, 2017), pp. 13–16.
[15]N. Gong, T. Idé, S. Kim, I. Boybat, A. Sebastian, V. Narayanan, and T. Ando, Nat. Commun. **9**, 2102 (2018).
[16]M. Cassinerio, N. Ciocchini, and D. Ielmini, Adv. Mater. **25**, 5975 (2013).
[17]C. D. Wright, P. Hosseini, and J. A. V. Diosdado, Adv. Funct. Mater. **23**, 2248 (2013).
[18]P. Hosseini, A. Sebastian, N. Papandreou, C. D. Wright, and H. Bhaskaran, IEEE Electron Device Lett. **36**, 975 (2015).
[19]A. Sebastian, T. Tuma, N. Papandreou, M. Le Gallo, L. Kull, T. Parnell, and E. Eleftheriou, Nat. Commun. **8**, 1115 (2017).
[20]G. W. Burr *et al.*, Adv. Phys. X **2**, 89 (2017).
[21]M. Le Gallo, A. Sebastian, G. Cherubini, H. Giefers, and E. Eleftheriou, in *International Electron Devices Meeting (IEDM)* (IEEE, 2017), pp. 28–23.
[22]S. Qaisar, R. M. Bilal, W. Iqbal, M. Naureen, and S. Lee, J. Commun. Networks **15**, 443 (2013).
[23]M. Le Gallo, A. Sebastian, R. Mathis, M. Manica, H. Giefers, T. Tuma, C. Bekas, A. Curioni, and E. Eleftheriou, Nat. Electron. **1**, 246–253 (2018).
[24]G. W. Burr *et al.*, IEEE Trans. Electron Devices **62**, 3498 (2015).
[25]S. Ambrogio, P. Narayanan, H. Tsai, R. M. Shelby, I. Boybat, C. Nolfo, S. Sidler, M. Giordano, M. Bodini, N. C. Farinha *et al.*, Nature **558**, 60 (2018).
[26]S. Nandakumar, M. Le Gallo, I. Boybat, B. Rajendran, A. Sebastian, and E. Eleftheriou, in *International Symposium on Circuits and Systems (ISCAS)* (IEEE, 2018), pp. 1–5.
[27]T. Gokmen and Y. Vlasov, Front. Neurosci. **10**, 333 (2016).
[28]P. Narayanan, A. Fumarola, L. Sanches, K. Hosokawa, S. Lewis, R. Shelby, and G. Burr, IBM J. Res. Dev. **61**, 1 (2017).
[29]S. Hochreiter and J. Schmidhuber, Neural Comput. **9**, 1735 (1997).
[30]J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, in *International Conference on Machine Learning* (2015), pp. 2067–2075.
[31]G.-Q. Bi and M.-M. Poo, J. Neurosci. **18**, 10464 (1998).
[32]B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, Proc. IEEE **102**, 699 (2014).
[33]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 *et al.*, Science **345**, 668 (2014).
[34]S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, Proc. IEEE **102**, 652 (2014).
[35]G. Indiveri and S.-C. Liu, Proc. IEEE **103**, 1379 (2015).
[36]K. Meier, in *International Electron Devices Meeting (IEDM)* (IEEE, 2015), pp. 4–6.
[37]M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, IEEE Micro **38**, 82 (2018).
[38]T. Tuma, A. Pantazi, M. Le Gallo, A. Sebastian, and E. Eleftheriou, Nat. Nanotechnol. **11**, 693 (2016).
[39]B. B. Averbeck, P. E. Latham, and A. Pouget, Nat. Rev. Neurosci. **7**, 358 (2006).
[40]W. Maass, Proc. IEEE **103**, 2219 (2015).
[41]S. Kim, M. Ishii, S. Lewis, T. Perri, M. BrightSky, W. Kim, R. Jordan, G. Burr, N. Sosa, A. Ray *et al.*, in *International Electron Devices Meeting (IEDM)* (IEEE, 2015), pp. 17–11.
[42]D. Kuzum, R. G. Jeyasingh, B. Lee, and H.-S. P. Wong, Nano Lett. **12**, 2179 (2012).
[43]B. L. Jackson, B. Rajendran, G. S. Corrado, M. Breitwisch, G. W. Burr, R. Cheek, K. Gopalakrishnan, S. Raoux, C. T. Rettner, A. Padilla *et al.*, ACM J. Emerging Technol. Comput. Syst. (JETC) **9**, 12 (2013).
[44]M. Suri *et al.*, in *International Electron Devices Meeting (IEDM)* (2011), pp. 4.4.1–4.4.4.
[45]S. Sidler, A. Pantazi, S. Woźniak, Y. Leblebici, and E. Eleftheriou, in *International Conference on Artificial Neural Networks* (Springer, 2017), pp. 281–288.
[46]T. Tuma, M. Le Gallo, A. Sebastian, and E. Eleftheriou, IEEE Electron Device Lett. **37**, 1238 (2016).
[47]S. Woźniak, T. Tuma, A. Pantazi, and E. Eleftheriou, in *IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE, 2016), pp. 365–368.
[48]A. Pantazi, S. Woźniak, T. Tuma, and E. Eleftheriou, Nanotechnology **27**, 355205 (2016).
[49]O. Bichler, M. Suri, D. Querlioz, D. Vuillaume, B. DeSalvo, and C. Gamrat, IEEE Trans. Electron Devices **59**, 2206 (2012).
[50]S. Thorpe, D. Fize, and C. Marlot, Nature **381**, 520 (1996).
[51]W. Maass, Neural Networks **10**, 1659 (1997).
[52]T. Moraitis, A. Sebastian, I. Boybat, M. L. Gallo, T. Tuma, and E. Eleftheriou, in *International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2017), pp. 1823–1830.
[53]T. Moraitis, A. Sebastian, and E. Eleftheriou, in *International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2018).
[54]F. Xiong, A. D. Liao, D. Estrada, and E. Pop, Science **332**, 568 (2011).
[55]G. Bruns, P. Merkelbach, C. Schlockermann, M. Salinga, M. Wuttig, T. Happ, J. Philipp, and M. Kund, Appl. Phys. Lett. **95**, 043108 (2009).
[56]M. Salinga, B. Kersting, I. Ronneberger, V. P. Jonnalagadda, X. T. Vu, M. Le Gallo, I. Giannopoulos, O. Cojocaru-Mirédin, R. Mazzarello, and A. Sebastian, Nat. Mater. **17**, 681–685 (2018).
[57]S. Kim, N. Sosa, M. BrightSky, D. Mori, W. Kim, Y. Zhu, K. Suu, and C. Lam, in *International Electron Devices Meeting (IEDM)* (IEEE, 2013), pp. 30–37.
[58]W. W. Koelmans, A. Sebastian, V. P. Jonnalagadda, D. Krebs, L. Dellmann, and E. Eleftheriou, Nat. Commun. **6**, 8181 (2015).
[59]W. Kim, M. BrightSky, T. Masuda, N. Sosa, S. Kim, R. Bruce, F. Carta, G. Fraczak, H. Cheng, A. Ray *et al.*, in *International Electron Devices Meeting (IEDM)* (IEEE, 2016), pp. 2–4.
[60]I. Boybat, M. Le Gallo, S. Nandakumar, T. Moraitis, T. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, and E. Eleftheriou, Nat. Commun. **9**, 2514 (2018).
[61]R. F. Freitas and W. W. Wilcke, IBM J. Res. Dev. **52**, 439 (2008).
[62]P. Cappelletti, in *International Electron Devices Meeting (IEDM)* (IEEE, 2015), pp. 10–11.
[63]L. Fiorin, R. Jongerius, E. Vermij, J. van Lunteren, and C. Hagleitner, IEEE Trans. Parallel Distrib. Syst. **29**, 115 (2018).
[64]N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, in *Proceedings of the 44th Annual International Symposium on Computer Architecture* (ACM, 2017), pp. 1–12.