

Review

Not peer-reviewed version

Exploiting Static Conductance and Dynamic Switching of Memristors for Artificial Intelligence Applications

[Zheng Miao](#) , Saitao Zhang , [Congcong Hong](#) , [Yongxiang Li](#) , Yubiao Luo , Shiqing Wang , [Zhong Sun](#) *

Posted Date: 23 March 2026

doi: 10.20944/preprints202603.1727.v1

Keywords: memristor; artificial intelligence; static conductance; dynamic switching; in-memory computing; analog matrix computing; neuromorphic computing



Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a [Creative Commons CC BY 4.0 license](#), which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Review

Exploiting Static Conductance and Dynamic Switching of Memristors for Artificial Intelligence Applications

Zheng Miao ^{1,2}, Saitao Zhang ^{1,2}, Congcong Hong ^{1,2}, Yongxiang Li ^{1,2}, Yubiao Luo ^{1,2}, Shiqing Wang ^{1,2} and Zhong Sun ^{1,2,3,*}

- ¹ Institute for Artificial Intelligence, Peking University, Beijing 100871, China
- ² School of Integrated Circuits, Peking University, Beijing 100871, China
- ³ Beijing Advanced Innovation Center for Integrated Circuits, Beijing 100871, China
- * Correspondence: zhong.sun@pku.edu.cn

Abstract

Memristors are resistive switching devices whose conductance states can be programmed by applied voltages. They exhibit two fundamental characteristics—static conductance and dynamic switching—both of which can serve as computing primitives to accelerate artificial intelligence (AI) algorithms. Computations based on these physical properties offer several key advantages, including massive parallelism, low latency, and reduced data movement. This Review integrates perspectives from device physics to system-level implementation, beginning with the fundamentals of memristive behavior and progressing to two primary application domains. First, we examine how static conductance characteristics are exploited in in-memory analog computing, particularly for matrix–vector multiplication and matrix equation solving, and discuss how these primitives enable efficient neural network inference and training. Second, we survey dynamic switching behaviors and their algorithmic applications, including stateful logic for digital in-memory acceleration, attractor networks for associative memory, reservoir computing and spatiotemporal signal processing using transient device dynamics, biologically inspired spike-timing-dependent plasticity, and stochastic computation. In addition, we discuss key challenges such as device variability, stochastic switching, interconnect parasitics, peripheral DAC/ADC overhead, and endurance limitations. We also highlight opportunities for future development, emphasizing algorithm–hardware co-design to leverage application-specific error tolerance and mitigate device non-idealities. Finally, we outline promising research directions aimed at realizing robust, scalable, and energy-efficient memristor-based AI systems.

Keywords: memristor; artificial intelligence; static conductance; dynamic switching; in-memory computing; analog matrix computing; neuromorphic computing

1. Introduction

The scale and computational demands of artificial intelligence (AI) models have grown explosively in recent years. In this work, AI is broadly defined to encompass a wide spectrum of computational paradigms, including traditional machine learning, deep neural networks, transformer-based foundation models, as well as neuromorphic and bio-inspired learning approaches. From large pre-trained foundation models to real-time inference on resource-constrained edge devices, modern AI workloads impose increasingly stringent requirements on computing power, memory bandwidth, and energy efficiency. However, conventional digital computing architectures, which rely heavily on sequential execution, struggle to meet these demands efficiently [1]. This challenge is further exacerbated by the slowdown of Moore’s Law, which has limited the performance gains achievable through transistor scaling [2]. Meanwhile, traditional von

Neumann architectures suffer from significant energy and latency overheads due to the frequent transfer of large volumes of data between physically separated processing and memory units [3–5]. Consequently, developing novel computing architectures that enhance parallelism and minimize data movement has become a critical necessity for next-generation AI systems.

Memristors—two-terminal electronic elements with programmable conductance—have emerged as promising building blocks for in-memory computing (IMC) and brain-inspired hardware [6–8]. As illustrated in Figure 1, their functionality is fundamentally characterized by two key properties: (i) static conductance—the ability to set and retain an analog (or multi-level) conductance that can represent matrix elements; and (ii) dynamic switching—nonvolatile or volatile, time-dependent resistive dynamics that enable the implementation of perceptron-like neuron behavior, short-term synaptic plasticity, stochastic transitions, and other temporal primitives.

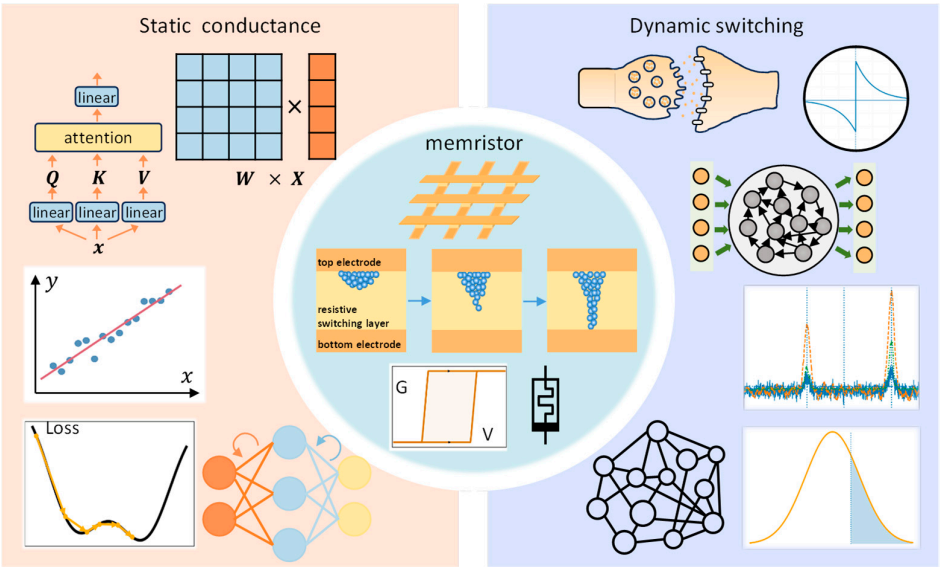


Figure 1. Memristor fundamentals and AI applications: physical mechanism, applications of static conductance in AI, and applications of dynamic switching in AI.

Exploiting the static conductance of memristors arranged in crossbar arrays, a primary application is analog matrix computing (AMC), which enables massively parallel, low-energy operations. The AMC paradigm encompasses two major pathways. The first is matrix-vector multiplication (MVM), the fundamental operation in fully connected, convolutional, and attention-based layers of neural networks [9]. The second pathway extends AMC to the direct, one-step solution of matrix equations [10]. This is achieved through closed-loop circuits that integrate crossbars with feedback amplifiers to perform matrix inversion, generalized inverse and related linear solving tasks in the analog domain.

Dynamic switching behavior refers to time-dependent resistive changes and switching events in memristors. Such dynamics enable a wide range of temporal and logic functions [11–14]. Stateful logic and attractor networks typically rely on nonvolatile memristors, as these applications require deterministic switching behavior and long-term retention. In contrast, devices exhibiting controlled decay, thresholding, or stochastic switching can function as compact physical neurons, implement local spike-timing-dependent plasticity (STDP), realize reservoir computing substrates, or serve as on-chip sources of randomness for probabilistic inference. In practice, hybrid designs that combine volatile temporal processors with nonvolatile readout or storage allow systems to integrate short-term dynamics with long-term weight storage.

This Review frames memristor-based AI hardware around these two complementary device capabilities—static conductance and dynamic switching—and systematically surveys physical mechanisms, key electrical metrics, representative system-level implementations, and remaining

challenges. After an overview of device physics and performance metrics (Section 2), we examine static-conductance approaches for parallel and analog matrix computations (Sections 3) and dynamic-switching approaches for stateful logic, attractor network, reservoir computing, temporal processing, STDP and stochastic computation (Sections 4). We conclude with a synthesis of open problems and a perspective on co-design strategies that bridge materials, circuits and algorithms. This review ultimately seeks to guide future efforts in developing robust, scalable, and application-tailored memristor-based AI systems.

2. Memristor Background

The memristor concept was first proposed by Leon Chua (1971) and later experimentally demonstrated in nanoscale two-terminal devices [15,16]. Fundamentally, a memristor encodes the history of applied electrical stimuli into a programmable resistance (or conductance) state. In practice, most hardware implementations adopt a metal–insulator–metal (MIM) “sandwich” structure, where a thin active layer—typically only a few nanometers thick—is positioned between the top and bottom electrodes. The composition of the active material and the engineered electrode interfaces determine the dominant switching mechanism and, consequently, the device’s electrical characteristics.

The switching physics of modern memristors can be broadly categorized into several well-established mechanisms. A major class comprises nonvolatile resistive switching processes, including the valence-change mechanism (VCM), electrochemical metallization (ECM), and phase-change memory (PCM). In VCM-based devices, resistive switching originates from the field-driven migration of oxygen vacancies in oxides such as HfO_2 or TaO_x , resulting in the formation and rupture of conductive filaments (CFs) under voltage bias [17–20]. Similarly, ECM devices operate through the migration and electrodeposition of active metal ions (e.g., Ag^+ or Cu^{2+}), which form or dissolve metallic conductive paths [21,22]. Notably, both VCM and ECM behaviors can be described within a unified voltage-driven ion migration framework, where defect or metal-ion motion governs the resistive states [3]. In PCM devices, resistance modulation arises from reversible phase transitions between amorphous (high-resistance) and crystalline (low-resistance) states in chalcogenide materials such as Ge-Sb-Te , triggered by Joule heating and rapid quenching [23,24]. These nonvolatile devices hold strong potential for high-density memory and in-memory computing applications.

In addition to nonvolatile devices, volatile or threshold-type memristors spontaneously return to a high-resistance state once the external stimulus is removed, making them well-suited for dynamic and temporal computing [25–27]. Representative examples include diffusion-mediated memristors, Ovonic threshold switching elements, and VO_2 -based Mott devices. Such volatility is particularly valuable for implementing short-term memory and temporal filtering in reservoir computing and compact neuronal models. These categories are not strictly distinct: material selection, interface engineering, current compliance, and pulse protocols can tune a device’s behavior along a continuum from volatile to nonvolatile on the same platform [6]. Therefore, practical classification is often application-driven. From this perspective, it is useful to adopt consistent terminology: in this review, static conductance refers to conductance states intended for long-term weight storage and computation, whereas dynamic switching refers to resistive state transitions exploited for stateful logic, attractor networks, temporal processing, and plasticity.

Key device performance metrics that determine suitability for AI workloads include: settling and retention times, programming linearity and symmetry, multilevel resolution, cycle-to-cycle and device-to-device variability, endurance, on/off contrast, switching energy, and the probability of stochastic switching under a given input pulse. Subsequent sections discuss how these physical characteristics are leveraged in specific circuit implementations and algorithmic frameworks.

3. Static Conductance of Memristors and Their Applications in AI

3.1. Matrix-Vector Multiplication Primitive on Crossbar Arrays

A memristor crossbar array serves as a fundamental building block for IMC, efficiently performing MVM [28–32]. As shown in Figure 2a, the array stores programmable conductances A_{ij} at each crosspoint, representing the weight matrix A . When a voltage vector $\mathbf{x}$ is applied to the word lines (WLs), the current flowing through each device is $I_{ij} = A_{ij} \cdot x_j$. According to Kirchhoff's current law (KCL), the total current in each bit line (BL) is the sum of the currents from all connected devices: $I_i = \sum_j G_{ij} v_j$. This current represents exactly the dot product between the i -th WL of the conductance matrix and the input vector. The resulting currents are then converted into a digital MVM result by peripheral circuitry, including transimpedance amplifiers (TIAs) and analog-to-digital converters (ADCs).

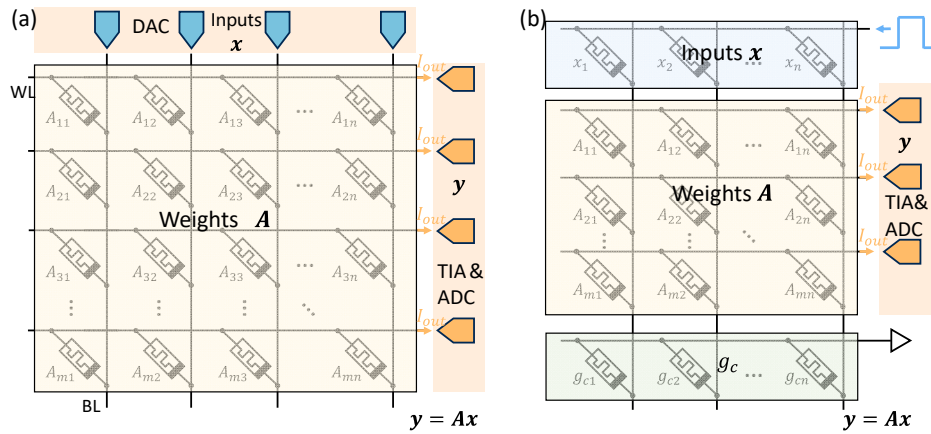


Figure 2. MVM with memristor crossbar arrays. (a) Crossbar MVM principle. Each weight element is encoded as a memristor conductance A_{ij} . Applying an input voltage vector on the WLs produces BL currents $I_i = \sum_j G_{ij} V_j$, which implement the MVM in one physical step. (b) Dual in-memory computation scheme. Inputs are encoded as conductances in another row of the array. Multiplication between weight and input is realized.

MVM executes n -dimensional dot products in a single physical step across a crossbar array, providing massive parallelism that can outperform serial digital implementations. IMC leverages this property to perform computation where the weights are stored, avoiding repeated data transfer between memory and processor. However, this crossbar-based implementation of MVM only partially alleviates the von Neumann bottleneck: the array typically stores only the weight matrix, while input values must still be supplied from external sources. To address this limitation, Wang *et al.* propose a dual in-memory computing method where the input vector is also represented as conductance states within the array, as shown in Figure 2b [33]. In this approach, a voltage pulse is applied to the word line corresponding to the row that stores the input vector $\mathbf{x}$. As a result, the bit-line (BL) voltages are modulated according to KCL. By introducing conductance compensation (g_c) to equalize the total conductance connected to each BL, a linear mapping between BL voltage and input value can be achieved: $V_{BLj} = kx_j$, where k is a constant. This approach enables both weights and inputs to be processed within the array, eliminating the need for digital-to-analog converters (DACs) for input encoding, thereby significantly reducing external data movement and improving array utilization.

This parallel MVM primitive, implemented at the physical device level, is a cornerstone for accelerating linear operators in a wide range of neural network architectures. However, in terms of precision, practical implementations have typically achieved effective resolutions of approximately 5–8 bits for specific tasks [34].

3.2. Applications of MVM in Neural Networks

A Memristor crossbars offer a versatile MVM primitive capable of accelerating a wide range of neural network architectures, and extensive research has demonstrated their practical integration.

Fully Connected Network (FCN). A fully connected layer computes $\mathbf{y} = \mathbf{W} \cdot \mathbf{x} + \mathbf{b}$, as illustrated in Figure 3a. Mapping this operation onto a crossbar is straightforward: each weight W_{ij} is programmed as a conductance A_{ij} , and the input vector $\mathbf{x}$ is applied as WL voltages; the resulting BL currents represent the output activations [35–39]. Because dense MVMs dominate many AI workloads, crossbars deliver high throughput by performing numerous dot products in parallel. Representative works span device-to-system considerations: integration studies combine arrays with CMOS peripherals to form programmable systems on a chip (SoC) that tolerate variability via calibration [35]; Dot-Product Engine-type architectures explore weight encoding, negative-value mapping, and precision decomposition [40]; and mixed-precision approaches perform most dot products in low-precision analog and correct residuals digitally, achieving efficient energy–accuracy tradeoffs [29]. FCNs represent a well-established and promising target for crossbar-based acceleration, with substantial progress already demonstrated in practice.

Convolutional Neural network (CNN). A convolution operation can be formulated as multiple local dot products between flattened kernels and input patches, as illustrated in Figure 3b [41]. Consequently, convolutions map naturally to memristor crossbars by storing flattened kernels as column-wise conductance vectors and applying input patches as row voltages [34,42–45]. The primary challenges lie in limited array sizes and efficient input reuse. Key studies have addressed these issues through weight encoding and precision-splitting schemes to reduce column replication, as well as optimized tiling and reuse strategies to improve array utilization. Moreover, several fully hardware-implemented CNN prototypes have demonstrated the feasibility of end-to-end on-chip inference [9,40,46]. Together, these results confirm that, with proper mapping and peripheral co-design, memristor crossbars can effectively accelerate convolutional layers.

Recurrent neural Network (RNN). Recurrent models reuse the same weight matrices across time steps, so the principal operations remain MVMs of the form $\mathbf{W} \cdot \mathbf{x}_t$ and $\mathbf{W} \cdot \mathbf{h}_{t-1}$ for the gates and candidate updates in an LSTM cell, as illustrated in Figure 3c [47]. Memristor crossbars can store the weight matrices for input and recurrent connections and be invoked at each time step to perform the required MVMs in parallel [48–52]. In practice, however, the time-recursive nature of RNNs raises two additional requirements: maintaining stable programmed conductances over repeated reads and writes, and controlling accumulated numerical errors across multiple time steps. Experimental and system-level studies have addressed these challenges: early demonstrations verified low-power LSTM inference, followed by in-situ training approaches that adapt to device nonidealities and system prototypes that incorporate calibration and mixed-precision correction [48,50]. Achieving high-precision conductance programming is particularly crucial to prevent long-term error accumulation.

Graph Neural Network (GNN). A prototypical graph convolutional layer (GCN) implements $\mathbf{H}(l+1) = \sigma(\hat{\mathbf{A}} \cdot \mathbf{H}(l) \cdot \mathbf{W}(l))$, where $\hat{\mathbf{A}}$ is an adjacency matrix and $\mathbf{H}(l)$ is the node-feature matrix of the l -th layer, as illustrated in Figure 3d [53]. Because adjacency matrices are sparse and access patterns irregular, directly mapping GNN operations onto dense crossbars results in low utilization and significant energy inefficiency [54,55]. Recent studies have proposed sparse-friendly mappings, communication support, and heterogeneous architectures that decouple the aggregation and dense transformation stages. For example, IMA-GNN organizes dataflow around graph aggregation, while ReGNN partitions dense and sparse computations for improved efficiency [56,57]. These works demonstrate that, with sparsity-aware scheduling and heterogeneous compute support, crossbar-based GNN accelerators can achieve competitive performance [58–60].

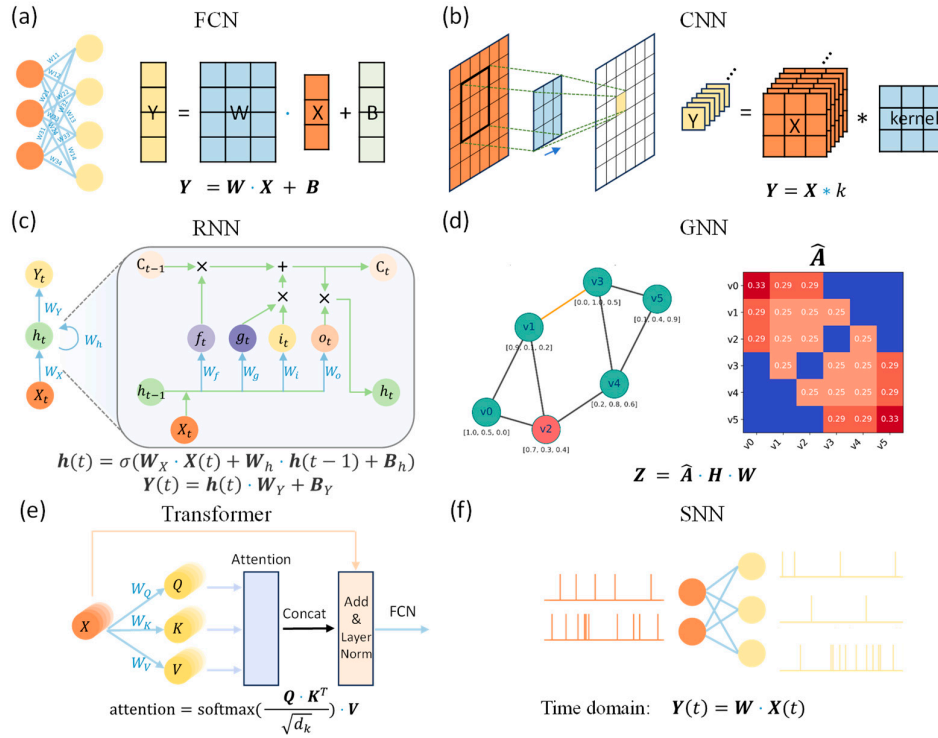


Figure 3. MVM mappings for common AI operators. (a) FCN: each output neuron performs a weighted sum over all inputs via crossbar MVM. (b) CNN: convolution as repeated MVMs over input patches and kernels; patch flattening maps the convolution to MVM operations. (c) RNN and LSTM: time-step matrix operations implemented by sequential MVMs and local state updates. (d) GNN: graph convolution and message passing realized by MVMs and aggregation. (e) Transformer: linear projections for Q, K, V and the attention-weighted sum require large numbers of MVMs. (f) SNN: temporal inputs are encoded as pulses and integrated by MVM-like accumulation over time.

Transformer. The computational bottleneck of Transformers lies in repeated dense matrix multiplications, prominently including the generation of Query (Q), Key (K), and Value (V) matrices through matrix multiplications of input embeddings. Subsequently, the scaled dot-product attention computes attention scores through another matrix multiplication (QK^T), followed by a weighted sum with V , expressed as $\text{Softmax}\left(Q \cdot \frac{K^T}{\sqrt{d_k}}\right) \cdot V$, as illustrated in Figure 3e [61]. Unlike many CNN kernels, the operands in attention layers vary dynamically at runtime, and generative inference requires storing and reusing key/value projections from previously generated tokens. Major challenges include minimizing frequent writes of intermediate results and reducing ADC/DAC overhead. Recent solutions address these issues through architectural and algorithmic co-design: ReTransformer employs matrix decomposition and pipelined execution; X-Former introduces sequence blocking for long-context processing; and analog in-memory attention approaches retain token projections in analog or hybrid storage to reduce data movement and latency during generation [62–64]. These works indicate that with co-optimized dataflow and approximation strategies, memristor crossbars can efficiently accelerate attention mechanisms in Transformer models [65–68].

Spiking Neural Network (SNN). Spiking networks compute weighted sums of sparse, temporally coded spike trains, meaning their synaptic operations can be expressed as dot products between spike-based inputs and conductance-encoded weights, as illustrated in Figure 3f. In addition, some volatile or diffusive memristors inherently exhibit thresholding and short-term dynamic behaviors, enabling compact hardware neuron implementations that integrate seamlessly with synaptic crossbars [69]. Architectures combining sensing, event encoding, spike generation, and

crossbar-based synapses minimize I/O overhead and enable highly event-driven computation pipelines. Several studies have also explored on-chip learning rules, such as STDP and its variants, implemented through local updates [70]. Overall, memristor hardware has demonstrated the ability to support both the synaptic MVMs and the compact neuronal primitives necessary for building dense, low-power SNN accelerators.

3.3. Analog Matrix Equation Solving Circuits

Memristor crossbars can also be employed to implement efficient analog matrix equation solvers [10,71]. By constructing closed-loop feedback topologies in which crossbars are embedded within the feedback paths of amplifiers—such as operational amplifiers (op-amps) or transimpedance amplifiers (TIAs)—these analog matrix computing (AMC) circuits can perform more advanced linear algebra operations relevant to AI applications. Figure 4 summarizes several representative closed-loop AMC circuits, including those for matrix inversion, pseudoinversion, eigenvector extraction, and sparse approximation.

Matrix Inversion (Figure 4a). The inversion circuit (INV) solves the linear system $\mathbf{A} \cdot \mathbf{x} = \mathbf{y}$ for an $n \times n$ matrix $\mathbf{A}$. The crossbar that encodes $\mathbf{A}$ is connected in a global feedback loop through op-amps. Thanks to the op-amp virtual-ground property, the input nodes are maintained at approximately 0 V during operation. When an input current vector $-\mathbf{y}$ is applied to the op-amp inputs, invoking KCL at the op-amp input gives

$$I_i = -y_i + \sum_j A_{ij} x_j = -y_i + (\mathbf{A} \cdot \mathbf{x})_i$$

Because the op-amp input draws negligible current in steady state, the closed loop enforces $-\mathbf{y} + \mathbf{A} \mathbf{x} = 0$, leading to the steady-state output

$$\mathbf{x} = \mathbf{A}^{-1} \cdot \mathbf{y}$$

Including op-amp dynamics using a single-pole model shows that the output evolves continuously from an initial condition toward this steady-state solution [72]. The transient behavior satisfies a first-order differential equation of the form

$$\frac{d\mathbf{x}}{dt} = \frac{1}{\tau} \mathbf{U} (\mathbf{A} \cdot \mathbf{x} - \mathbf{y})$$

where τ is the circuit time constant and $\mathbf{U}$ is a diagonal scaling matrix. The convergence time of this topology depends on the spectral properties of $\mathbf{A}$ (in particular its smallest eigenvalue), rather than directly on matrix size [72]. Specifically, a larger minimum eigenvalue (or equivalently, a smaller condition number) leads to faster convergence. This spectral dependence explains why closed-loop AMC can achieve very fast solutions for well-conditioned matrices.

Generalized Inverse (Figures. 4b–c). Non-square or rank-deficient problems require a generalized inverse (GINV). AMC can realize both left and right GINVs with paired crossbars and amplifier networks.

Figure 4b shows the left GINV circuit for the overdetermined case $m > n$. Two crossbar arrays store copies of $\mathbf{A}$ and two sets of amplifiers form the feedback network. When the input current vector $\mathbf{y}$ is applied, the first crossbar and its TIAs produce WL voltages $\mathbf{z}$ proportional to the residual $(\mathbf{y} - \mathbf{A}\mathbf{x})$:

$$\mathbf{z} = -\alpha (\mathbf{y} - \mathbf{A} \cdot \mathbf{x})$$

Applying KCL on the second crossbar column lines yields

$$\mathbf{A}^T \cdot (\mathbf{y} - \mathbf{A} \cdot \mathbf{x}) = 0$$

So the steady-state solution is

$$\mathbf{x} = (\mathbf{A}^T \cdot \mathbf{A})^{-1} \cdot \mathbf{A}^T \cdot \mathbf{y} = \mathbf{A}^+ \cdot \mathbf{y}$$

Figure 4c implements the right GINV for the underdetermined case $m < n$. Here the crossbars are programmed with $\mathbf{A}^T$ and the topology is modified so that KCL yields

$$-\mathbf{y} - \alpha \mathbf{A} \cdot \mathbf{A}^T \cdot \mathbf{z} = 0 \Rightarrow \alpha \mathbf{z} = -(\mathbf{A} \cdot \mathbf{A}^T)^{-1} \cdot \mathbf{y}$$

and back-substitution gives

$$\mathbf{x} = -\alpha \mathbf{A}^T \cdot \mathbf{z} = \mathbf{A}^T \cdot (\mathbf{A} \cdot \mathbf{A}^T)^{-1} \cdot \mathbf{y} = \mathbf{A}^+ \cdot \mathbf{y}$$

Both circuits compute minimum-norm solutions through fully analog feedback dynamics, which is useful for least-squares regression tasks in AI.

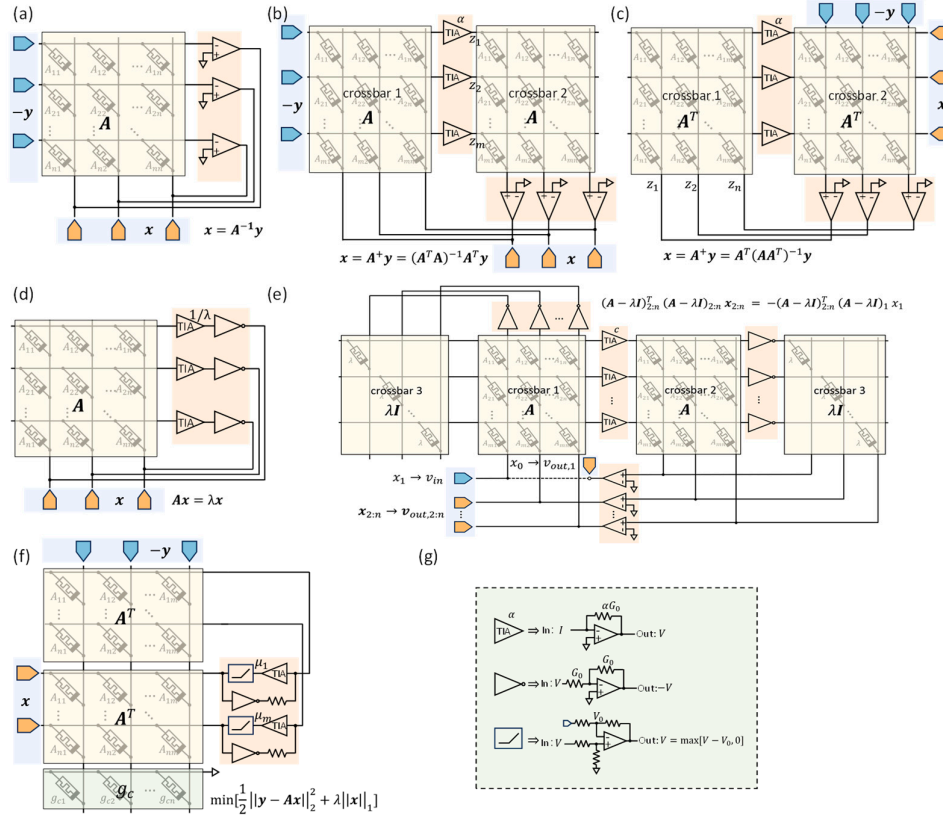


Figure 4. Closed-loop AMC circuits. (a) Matrix inversion circuit: crossbar coupled to global feedback via op-amps to solve $A \cdot x = y$ in the analog domain. (b) Left GINV circuit: two-array/feedback configuration yields $x = (A^T \cdot A)^{-1} \cdot A^T \cdot y$. (c) Right GINV circuit: topology using transposed matrix and feedback to compute $x = A^T \cdot (A \cdot A^T)^{-1} \cdot y$. (d) Eigenvector circuit: feedback configuration that enforces $A \cdot x = \lambda x$ to obtain eigenvectors for a chosen eigenvalue λ . (e) Modified-GINV circuit: two-array configuration derived from GINV topology to extract all eigenpairs. (f) Sparse approximation circuit: achieving one-step convergence to the sparse approximation solution. (g) Analog modules used across panels: TIA, analog inverter, and soft-threshold module.

Eigenvector Circuit (Figure 4d). If a TIA conductance encodes the derivative associated with an eigenvalue λ , the closed-loop dynamics enforce the eigenvalue equation

$$A \cdot x = \lambda x$$

By KCL, currents leaving each row correspond to $A \cdot x$; after conversion and scaling the circuit implements $A \cdot x = \lambda x$ [73]. The steady-state outputs are the eigenvector associated with λ . This analog approach enables direct extraction of principal directions, which can accelerate PCA and spectral methods used in AI pipelines [74].

Modified-GINV Circuit (Figure 4e). Fig. 4e illustrates an AMC circuit designed to solve for all eigenpairs (eigenvalues and eigenvectors) of a matrix [75]. This design, derived from the GINV circuit topology, operates by disconnecting one column feedback path and mapping the target matrix A and the diagonal matrix λI onto two distinct memristor arrays. Leveraging KCL, Ohm's law, and the virtual short and virtual ground principles of op-amps, the circuit inherently satisfies the following matrix relation:

$$(A - \lambda I)_{2:n}^T (A - \lambda I)_{2:n} x_{2:n} = -(A - \lambda I)_{2:n}^T (A - \lambda I)_1 x_1$$

In this formulation, the vector $(x_1, x_2, \dots, x_n)^T$ corresponds to the eigenvector associated with the current eigenvalue parameter λ . By sweeping the conductance values of the diagonal array to vary λ and monitoring the output voltage of the op-amp in the disconnected branch for a characteristic peak, the circuit can identify when λ coincides with an actual eigenvalue of $\mathbf{A}$. This approach enables the extraction of the complete set of eigenpairs for a given matrix, which can accelerate various AI applications that rely on eigendecomposition, including principal component analysis (PCA), spectral clustering and graph learning.

Sparse Approximation (Figure 4f). Figure 4f shows an AMC circuit that solves a sparse approximation (LASSO-like) problem:

$$\min_{\mathbf{x}} L(\mathbf{x}) = \min_{\mathbf{x}} [1/2 \|\mathbf{y} - \mathbf{A} \cdot \mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_1]$$

where λ controls the sparsity penalty [76]. In this design, the compensation conductance g_c is applied to let the conductance sum of each column equal to a constant, thus each column-line voltage is set by the mismatch $-\mathbf{y} + \mathbf{A} \cdot \mathbf{x}$, yielding a crossbar output current proportional to $\mathbf{A}^T (\mathbf{A} \mathbf{x} - \mathbf{y})$. That current is converted by TIAs to node voltages $\boldsymbol{\mu}$, and the sparsity outputs $\mathbf{x}$ is obtained through soft-thresholding nonlinearity (implemented by an analog threshold module, as shown in Figure 4g). Element-wise thresholding produces the solution vector $\mathbf{x}$ via a simple nonlinear map:

$$\mathbf{x} = \max[\boldsymbol{\mu} - \lambda, 0]$$

which acts as a hardware proximal operator for the L1 term under a nonnegativity assumption. The analog inverters are applied for cancelling the diagonal contribution of the Gram product $\mathbf{A}^T \cdot \mathbf{A}$. The circuit performs the locally competitive algorithm (LCA):

$$\frac{d\boldsymbol{\mu}}{dt} = \frac{1}{\tau} [-\boldsymbol{\mu} + \mathbf{A}^T \cdot \mathbf{y} - (\mathbf{A}^T \cdot \mathbf{A} - \mathbf{I}) \cdot \mathbf{x}]$$

This direct analog implementation of the differential equation can accelerate sparse coding, compressed sensing, and other AI primitives relying on L1 regularization.

These closed-loop AMC circuits convert physical parallelism and feedback dynamics into direct, analog solvers for linear-algebra problems central to AI. Thereby, they provide efficient hardware primitives for tasks such as regression, optimization, and neural network training.

3.4. Applications of Analog Matrix Equation Solving Circuits

3.4.1. Linear/Logistic Regression

As a fundamental machine learning (ML) model, linear regression is widely used for regression and predictive analysis across diverse fields, including biology, statistics and finance [77]. The GINV circuit described above can be directly applied to perform linear regression in a single step and has been demonstrated on standard benchmark datasets such as the Boston Housing dataset [78]. This approach enables one-step regression with comparable accuracy to digital baselines and offers a dramatic speedup and energy reduction owing to its fully parallel analog implementation [78]. In contrast to linear regression, which is a fully linear model, logistic regression introduces a nonlinear sigmoid function into the feature-to-output mapping to produce binary outcomes. As a widely used model for object classification and pattern recognition, logistic regression can be regarded as a single-layer feedforward neural network. It is also commonly employed as the final output layer in multilayer neural networks, with its training process equivalent to that of a single-layer network. Through the logit transformation, the logistic regression problem can be reduced to a linear regression problem. Based on this simplification, as shown in Figure 5a, the aforementioned linear regression (or GINV) circuit has been shown to implement logistic regression in one step—for example, in the training of a two-layer neural network for MNIST digit recognition and can effectively accelerate reservoir computing training [78–80]. Previous studies have also demonstrated that such circuits exhibit significant performance advantages across multiple benchmark metrics [81–85]. Their computational speed can further be enhanced through parameter co-optimization [86].

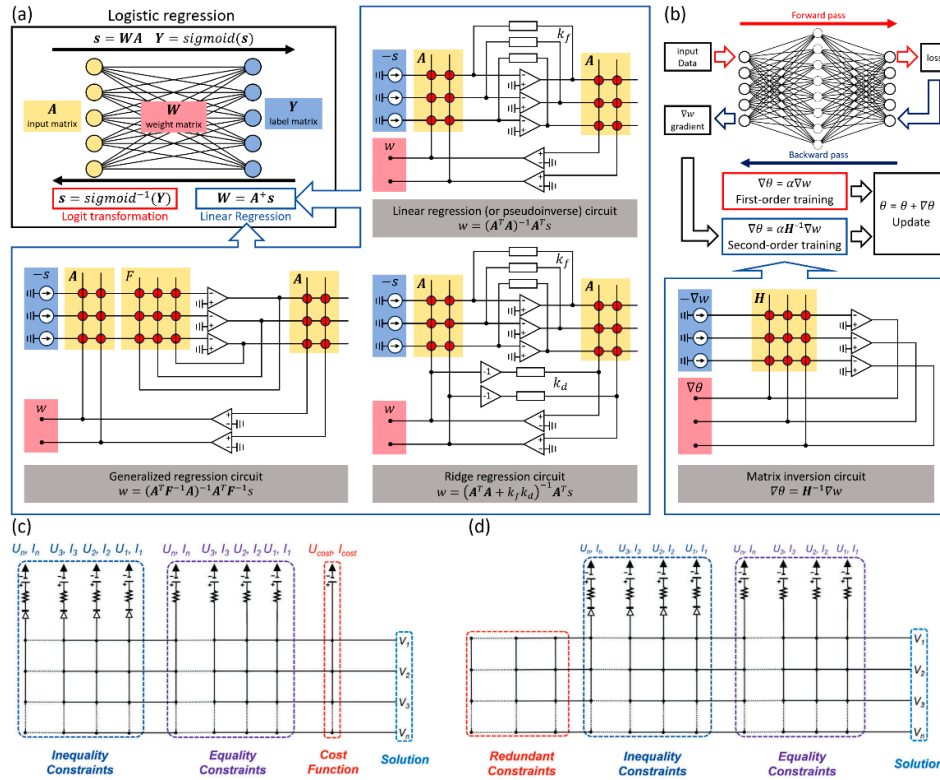


Figure 5. Applications of AMC Circuit. (a) Based on the logit transformation, logistic regression can be transformed into linear regression and accelerated through linear regression circuit. Generalized regression and ridge regression circuits can further expand the scope of accelerated regression problems. (b) The second-order training algorithm can adjust the optimization direction by inverting the SOI matrix to achieve stronger optimization effects than the first-order optimization algorithm. The matrix inversion circuit can effectively accelerate the SOI matrix inversion, thereby solving the problem of huge matrix inversion overhead in second-order optimization algorithms. (c) Mapping constrained LP problem to optimization circuits. (d) Mapping constrained QP problem to optimization circuits. Both the schematic of the LP and QP optimization circuit are demonstrated using a compact notation where a dot represents each memristor. Reproduced with permission of Ref. [105], Copyright [2024] Advanced Functional Materials.

Building on the basic GINV circuit structure, a generalized regression circuit has also been designed by replacing scalar conductances with a conductance array. The introduced matrix enables encoding additional information on data covariance for generalized linear regression or circuit preconditioning when solving linear systems [87]. Meanwhile, a ridge regression circuit has been developed to implement the regularization term by adding an additional negative feedback branch [88]. Although applications of generalized and ridge regression circuits in neural network training have not yet been reported, these designs have significantly broadened the functional scope of regression circuits and are expected to enable fully on-hardware neural network training [89].

3.4.2. Second-Order Neural Network Training

In deep neural network (DNN) training, second-order optimization methods converge much faster than first-order ones because they leverage the inversion of the second-order information (SOI) matrix to determine more accurate descent directions and step sizes [90]. However, on conventional computing platforms such as GPUs and CPUs, handling the massive SOI matrix incurs substantial computational and memory overhead. As illustrated in Figure 5b, the matrix inversion circuit described earlier can efficiently accelerate SOI matrix inversion, offering $O(1)$ computational

complexity and high parallelism. Analysis has shown that, compared with GPUs and PipeLayer (a framework for large-scale DNNs), the second-order training accelerator based on RRAM matrix inversion circuits achieves an average speed-up of $115.8\times/11.4\times$ and an average energy efficiency improvement of $41.9\times/12.8\times$ [91]. In parallel, inspired by the intrinsic mechanisms of analog matrix computing, the thermodynamic approach to computation has emerged as a promising paradigm for solving mathematical problems. Its core principle lies in exploiting the stochastic dynamic behavior of physical systems governed by the interplay of conservative, dissipative, and fluctuating forces [92]. Thermodynamic computing has been shown to accelerate tasks such as matrix determinant evaluation, and matrix equations solving [93,94]. Furthermore, it has been applied to Bayesian inference and to accelerating various second-order neural network training algorithms, including natural gradient descent and Kronecker-factored approximate curvature (KFAC) [95–97]. Notably, these thermodynamic approaches have demonstrated superior speed and energy efficiency compared with first-order methods such as stochastic gradient descent (SGD) and Adam.

3.4.3. Linear Programming

The parallelism of AMC circuits, combined with feedback-loop architectures, enables their widespread application in optimization tasks, including linear programming (LP) and quadratic programming (QP). LP and QP are extensively used in numerous scientific and engineering fields, such as machine learning, robot kinematics, drone control, resource allocation, and digital finance [98]. Currently, industry relies primarily on digital solvers, including optimized versions of the simplex and interior-point methods, to solve LP and QP problems. However, as matrix sizes grow, digital computers face increasing difficulty in solving these problems efficiently, while real-time applications demand faster solution speeds. LP and QP problems can be formulated as minimizing an objective function $\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{c}^T \mathbf{x}$ subject to equality and inequality constraints, where a QP problem reduces to an LP problem when the $\mathbf{Q}$ matrix is zero. Both the interior-point and simplex methods are iterative, with each iteration requiring multiple complex matrix computations, which imposes a lower bound on computational complexity for digital solvers [99,100]. AMC circuits provide an alternative approach by directly accelerating these matrix operations. For example, using AMC circuits to implement iterative methods such as the Alternating Direction Method of Multipliers can reduce the effective computational complexity of QP problems to approximately $O(k)$, where k is the iteration times that is typically small. Furthermore, optimized circuit structures can extend AMC solvers to tackle more complex problems, including mixed-integer programming [101,102].

By fully exploiting the inherent differential characteristics of feedback loops, several studies have proposed LP solvers based on recurrent neural network architectures and direct analog solvers [103–107]. As shown in Figures 5c and 5d, the ultra-fast hybrid computing system integrates tuning, control, optimization, reading circuits, and memristor arrays. By directly mapping the problem constraints and objective function to the memristor array conductances and input voltage values, the solution can be obtained in a single step at the output port, reducing the computational complexity of LP and QP optimization to $O(1)$. Experimental testing has shown that this system can solve optimization problems within 1 μs . In application scenarios such as miniature aerial vehicle control and edge computing for clinical image analysis, the memristor-based analog LP and QP problem solver demonstrates ultra-fast response speeds and energy efficiency improvements of over seven orders of magnitude, substantially alleviating computational pressure in these domains.

3.4.4. PCA and Spectral Clustering

PCA is a classical linear dimensionality reduction technique in machine learning. It relies on the eigendecomposition of data covariance matrices to extract eigenvectors (principal components) representing directions of maximum variance, with corresponding eigenvalues quantifying the information content. Conventional digital processors implement PCA using singular value decomposition or iterative algorithms, incurring $O(n^3)$ computational complexity for high-dimensional data and suffering from significant data movement overhead. The memristor-based

AMC circuit in Figure 4e provides a hardware acceleration solution: by mapping the covariance matrix $\mathbf{C}$ onto array conductances and sweeping the λ value in the diagonal array, the circuit solves all eigenpairs directly in the analog domain [75,108]. When λ coincides with an eigenvalue, the op-amp output exhibits a characteristic peak, while the voltage vector $(x_1, x_2, \dots, x_n)^T$ directly encodes the corresponding eigenvector. This analog-domain direct solution eliminates the need for iterative convergence and data movement, enabling PCA with ultra-low latency and energy consumption. The circuits have been validated on the Iris dataset and Wine dataset, with extracted principal components closely matching theoretical values.

Spectral clustering is a graph-based clustering algorithm that relies on eigendecomposition of the graph Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ [109]. Its core step extracts eigenvectors corresponding to the smallest nonzero eigenvalues as node spectral embeddings, followed by k -means clustering in the embedding space to partition graph nodes. For large-scale graphs with tens of thousands of nodes (e.g., social networks, knowledge graphs), Laplacian eigendecomposition imposes enormous computational demands that conventional digital processors struggle to meet for real-time applications. The AMC circuit in Figure 4e provides efficient hardware acceleration: by mapping the Laplacian matrix onto memristor arrays, the circuit completes eigendecomposition in constant time, directly outputting all required eigenvectors. Because the circuit solves the complete set of eigenpairs rather than only the dominant ones, it is particularly suited for spectral clustering, which requires multiple non-dominant eigenvectors corresponding to the smallest eigenvalues. The parallelism and low-latency characteristics of analog computing, combined with memristor in-memory processing, significantly alleviate the memory wall bottleneck in large-scale graph analytics, offering viable acceleration for real-time graph clustering, community detection in social networks, and image segmentation applications [110–112].

4. Dynamic Switching of Memristors and Their Applications in AI

Driven by an applied voltage, memristors change their conductance, exhibiting distinct conductance states. The high-conductance state (HCS) functions like a closed switch (logic 1), while the low-conductance state (LCS) acts as an open switch (logic 0). Beyond these static conductance characteristics, the dynamic switching process also enables various AI-related functions, allowing the devices to serve simultaneously as nonvolatile memory elements and *in situ* logic units.

4.1. Stateful Logic and In-Memory Logic Acceleration

Stateful logic executes Boolean operations directly within a memory array, by exploiting the abrupt switching process of memristor to indicate logic output [113–118]. It uses only the two conductance levels HCS and LCS to encode logic 1 and 0, respectively. Under proper voltage stimuli, SET and RESET transitions change a device to HCS or LCS, respectively. In this way, a memristor both stores a bit and participates in logic operation, eliminating costly data movement between separate memory and compute units.

Stateful logic gates operate by exploiting differences in conductance among input memristors to control intermediate node voltages. Those intermediate voltages, in turn, determine whether an output memristor receives a voltage that exceeds its switching threshold and therefore switches state. Different input patterns produce different node voltages and thus different outputs for the same physical connection pattern.

Material implication (IMP) was the earliest demonstrated stateful primitive for memristors [113]. As shown in Figure 6a, in the IMP circuit two memristors, A and Y, together with a load resistor are driven by two voltages: a conditioning voltage V_{con} applied to A and a SET voltage V_{set} applied to Y. The intermediate node voltage V_p is determined by KCL:

$$V_p = \frac{V_{\text{con}}G_A + V_{\text{set}}G_Y}{G_X + G_Y + G_L}$$

Here G_A and G_Y are the conductances of A and Y, and G_L denotes the load conductance. The logic design ensures that Y is SET (written to 1) only when A = 0 and Y = 0 initially; when A=1 and

Y=0, the node voltage remains below the SET threshold, preventing switching. If Y is already 1, the output stays unchanged. The final state of Y implements the IMP result. IMP, together with FALSE logic gate, can be composed to realize universal logic.

Memristor-Aided LoGIC (MAGIC) is a memristor-only logic family that mainly uses RESET transitions to implement gates such as NOR [114]. As shown in Figure 6b, input memristors A and B are preprogrammed to input values, and the output memristor Y is initially set to HCS. During evaluation, input lines are driven with voltages such that the intermediate node voltage V_P becomes

$$V_P = \frac{V_A G_A + V_B G_B}{G_A + G_B + G_Y}$$

If either input is logic 1, V_P shifts and the voltage across Y exceeds its RESET threshold, forcing Y to LCS (logic 0). Only if both inputs are 0, Y remains in HCS (logic 1). With NOR as a primitive, arbitrary Boolean logic can be synthesized in place within the array.

Stateful logic can be extended to the so-called stateful neural network framework [115]. As shown in Figure 6c, memristors A and B encode the logic inputs, and the output memristor Y stores the computation result. Applying KCL at an intermediate node, the voltage across the output memristor can be expressed as a weighted sum of input contributions:

$$V_Y - V_P = w_A x_A + w_B x_B + b$$

Here x_A and x_B denote input bits stored in memristor states, and w_A , w_B , b are effective weight terms derived from conductances and voltage biases. Because the output memristor has a switching threshold, the architecture naturally implements a nonlinear activation: if the weighted sum exceeds threshold, the output switches to 1; otherwise it remains 0. In this way, linear accumulation and a threshold activation are realized *in situ*, enabling implementation of single-layer perceptron for linearly separable logic functions, and, by cascading, multi-layer networks capable of XOR/XNOR and other linearly inseparable functions.

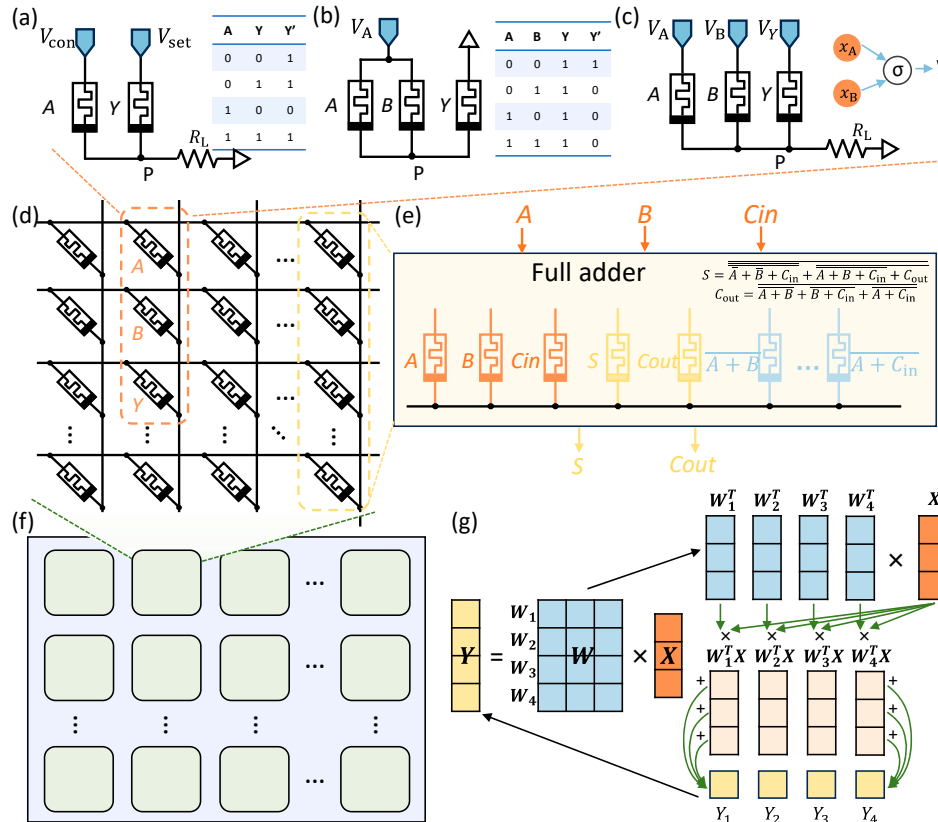


Figure 6. Stateful logic with memristors and its AI applications. (a) IMP primitive implemented with memristors. (b) MAGIC circuit for NOR. (c) Stateful neural network circuit. (d) Mapping between crossbar columns and

stateful gate execution. (e) 1-bit FA realized from NOR-based stateful gates mapped to a single crossbar column. (f) Architecture of multiple crossbar memory blocks. (g) MVM scheme adapted to stateful logic for accelerating neural-network primitives.

Using combinations of stateful gates, one can build digital adders, multipliers, and full arithmetic units inside memristive arrays. For example, a 1-bit full adder can be mapped to a column of memristors and a sequence of stateful operations, as shown in Figures 6d and 6e. Researchers have proposed MAGIC-based designs for fixed-point multiplication and for floating-point multiplication adapted to crossbar arrays [119]. Such adders and multipliers can be composed to implement MVM primitives, which are central to neural network inference and training. Complex nonlinear functions and gradients required by learning algorithms can be approximated by Taylor expansions and implemented using the same basic arithmetic primitives.

Several stateful processing-in-memory (PIM) architectures are built upon these MAGIC-based stateful logic primitives to accelerate AI workloads. As shown in Figures 6f and 6g, FloatPIM is a representative architecture that decomposes high-precision operations into sequences of bitwise stateful gates and maps them efficiently onto crossbar arrays [120]. FloatPIM and related systems implement convolution and fully connected layers by bit-serial or hybrid mappings, trading the number of bitwise cycles for reduced data movement and high array parallelism [119–123]. Other architectures—ConvPIM, RIME, ReHy_A, and DUAL—apply similar principles for CNNs, transformers, clustering, and other ML tasks [126,127]. Improvements in mapping, gate synthesis, and compiler-level scheduling are critical to achieve high throughput. On the other hand, hybrid analog–logic approaches may also be applied by combining analog crossbar MVM for bulk multiply–accumulate with stateful digital logic for activations, normalization, and control, thus maximizing the strengths of both paradigms [128].

4.2. Attractor Network

Beyond utilizing the inherent unidirectional threshold behavior of memristor, their SET/RESET switching characteristics can also be leveraged simultaneously. This manifests as a significant hysteresis loop in the conductance–voltage (G–V) plot. Building on this observation, Li *et al.* first proposed that memristor intrinsically function as an artificial neuron, providing a hysteretic nonlinear activation function [129]. Furthermore, they proved that a circuit comprising a column of memristor connected to a grounded resistor (Figure 7a) constitutes an attractor network, with the externally applied voltage defining an antisymmetric weight matrix (Figure 7b). This network operates via KCL, enabling recursive behavior through device interactions. Specifically, each device’s state change couples dynamically to all others, and a resistance shift in one device generates circuit feedback that propagates changes throughout the network.

Attractor networks are primarily employed for associative memory, wherein attractors store memory content. As the network stabilizes, other states converge towards these corresponding attractors, achieving the associative memory recall. By formulating an energy function for memristor network and analyzing its properties, this work demonstrated that state transitions require overcoming an energy threshold (Figure 7c). This enables the network to store more attractors (a fraction of 2^N), surpassing the linear capacity scaling in current Hopfield networks [11,130–132]. Furthermore, previous implementations typically require specialized circuit designs and utilize relatively expensive amplifier circuits as neurons, suffering significant latency and power consumption penalties. In contrast, memristor network comprises simple memristor arrays, where feedback operates continuously under KCL and accelerates attractor dynamics convergence.

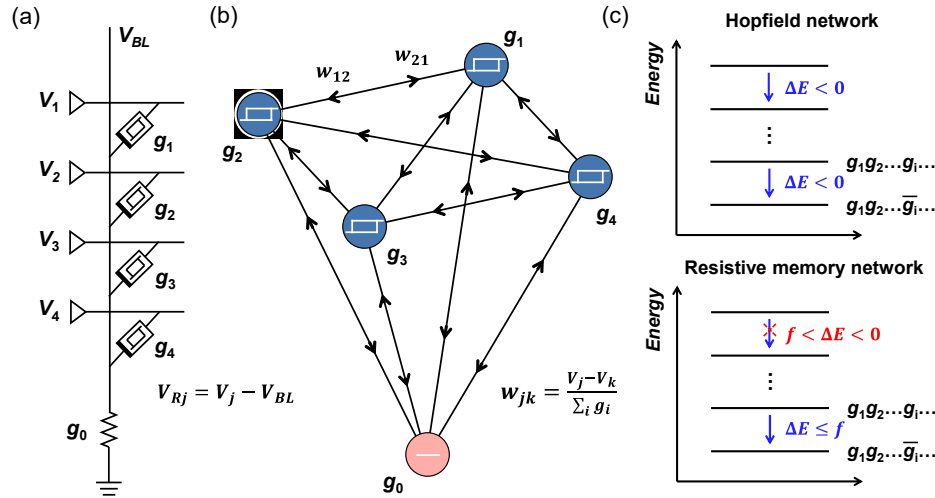


Figure 7. Attractor Network (a) Resistive-memory column circuit with a memristor and load resistor. (b) Abstracted RNN model of the memristor circuit, showing the weight definition. (c) Illustration of conditions for a 1-bit flip in the memristor network and comparison to a Hopfield network. Reproduced with permission of Ref. [129], Copyright [2024] Nature Communications.

4.3. Reservoir Computing and Spatiotemporal Signal Detection with Dynamic Memristors

Volatile memristors are particularly well suited for temporal and spatiotemporal processing. Their internal ionic or defect dynamics provide a controllable fading memory and intrinsic nonlinearity. The fading memory originates from migration and relaxation processes, which cause device responses to decay over characteristic time scales [79,133,134]. As a result, recent inputs strongly influence the current state, while the impact of older inputs gradually diminishes. This combination of short-term memory and nonlinear transduction makes dynamic memristors natural candidates for hardware reservoir computing [135]. They are also effective for local spatiotemporal feature extraction in edge sensors.

A memristor-based reservoir maps time-varying inputs into a high-dimensional dynamical state. Only the readout layer is trained, keeping training simple and efficient. In hardware, the reservoir is formed by a set of volatile memristors. Their transient conductance changes encode the recent history of input pulses. Nonlinearity in device I-V characteristics and threshold-like switching broaden the feature space. This improves separability of temporal patterns. Practical reservoir designs exploit these device-level properties together with input encoding and simple readout structures. They can perform classification, forecasting, and event detection in real time.

Experimental validation has employed arrays of 88 dynamic memristors to build a compact reservoir. They encoded input sequences as pulse trains and used the devices' transient responses as reservoir states. A linear readout was trained to classify framed MNIST patterns and to predict nonlinear time series, with the same ionic device dynamics simultaneously providing a fading memory of recent inputs and a nonlinear projection of temporal features. These are essential for framed-digit classification and time-series prediction in a small hardware footprint (Figures 8a and 8b) [136]. To further enhance state richness and resource efficiency, subsequent work combined simple masking with parallel device sampling [137]. This masked-parallel scheme generates virtual nodes from a limited number of physical devices. This approach achieved strong results on spoken-digit recognition and chaotic-series prediction (Figures 7c-7h). They later demonstrated fully analog reservoir stacks with non-volatile memristor readouts. These systems enabled low-power, real-time spatiotemporal processing such as ECG and gesture recognition [138].

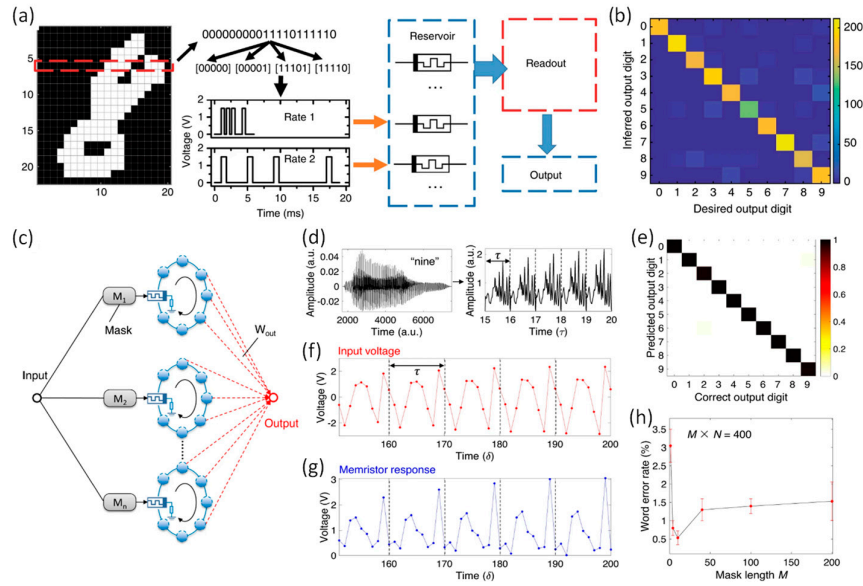


Figure 8. Reservoir computing (a–b) Schematic of a dynamic memristor reservoir system: input frames are encoded into pulse sequences and applied to the reservoir; framed-digit classification results are shown. Reproduced with permission of Ref. [136], Copyright [2017] Nature Communications. (c–h) Masking and parallel single-device architecture for state expansion: controlled masking, input scaling and feedback produce tunable state richness; spoken-digit recognition and Hénon-map prediction results are shown. Reproduced with permission of Ref. [137], Copyright [2021] Nature Communications.

Beyond pure RC, dynamic memristors have been integrated into adaptive neuromorphic perception pipelines [139–147]. These systems perform local spatiotemporal feature extraction and fast online adaptation. In these systems, memristor arrays implement tunable temporal kernels and rapid state updates that directly influence sensor-level decisions. For example, a differential neuromorphic pipeline was developed that uses memristor dynamics for near-sensor perception [140]. Their system enables fast adaptation in unstructured tasks and robust tactile and scene-level decision extraction in robotic scenarios (Figure 9), achieving low-latency responses by exploiting near-sensor processing that reduces data movement.

4.4. Spike-Timing-Dependent Plasticity

STDP is a local, timing-based synaptic learning rule [148,149]. It adjusts synaptic efficacy according to the relative timing of pre- and post-synaptic spikes. In the commonly used pair rule, the weight change follows

$$\Delta w = A_+ \exp(-\Delta t/\tau_+) \text{ for } \Delta t > 0$$

$$\Delta w = -A_- \exp(\Delta t/\tau_-) \text{ for } \Delta t < 0,$$

where $\Delta t = t_{pre} - t_{post}$. Positive Δt yields potentiation (LTP), and negative Δt yields depression (LTD) [148].

Hardware demonstrations show that memristors can implement STDP using simple spike protocols [150–159]. The first hardware evidence of memristive STDP was reported using a nanoscale memristor acting as a synapse in a hybrid system combining CMOS integrate-and-fire neurons with memristive synapses. In this system, the neuron circuit converted relative spike timing into pulse-width information to induce STDP [151]. Building on this foundation, passive memristor crossbars were integrated with analogue leaky-integrate-and-fire (LIF) silicon neurons to experimentally measure the canonical pair-STDP window (Figure 10a) [160]. This work establishes a practical route for device-level STDP in dense arrays. Pair-based STDP cannot capture all biological phenomena. Triplet-STDP models extend the rule by including higher-order spike interactions or trace variables. Devices with additional internal dynamics—so-called second-order memristors—can reproduce

triplet effects. Device-level triplet-STDP was demonstrated with weight updates under three-spike protocols (Figure 10b) [161]. The observed behavior matches triplet model predictions and highlights the role of internal device traces in shaping plasticity.

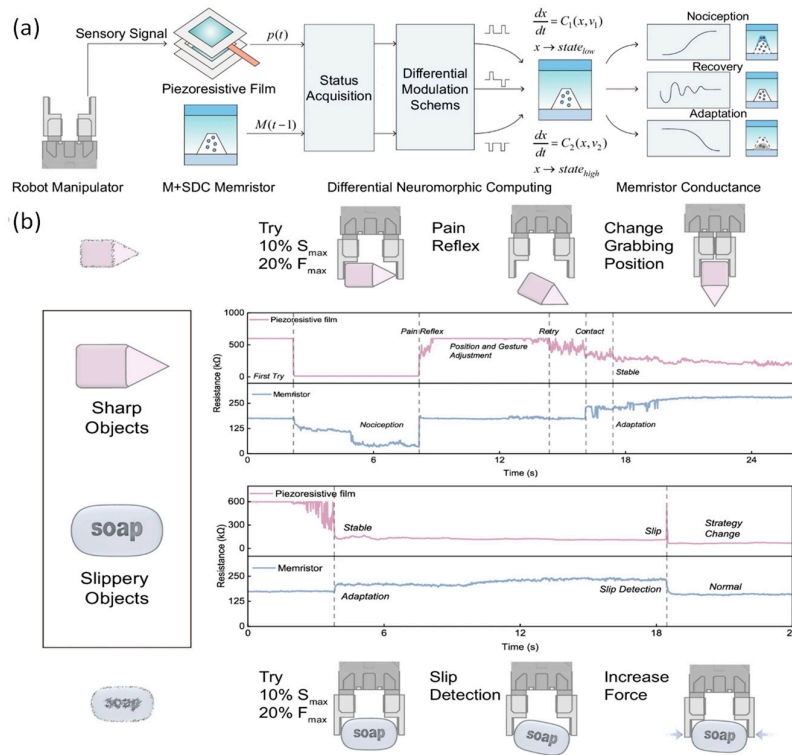


Figure 9. spatiotemporal signal detection (a) Memristor-based near-sensor processing and adaptive response pipeline. (b) Representative task demonstrations showing grasping/scene extraction outcomes. Reproduced with permission of Ref. [140], Copyright [2024] Nature Communications.

At system scale, STDP has been used for online learning and pattern tracking. Memristive synapses were integrated with simple spiking neurons [162]. Continuous online experiments were conducted in which synaptic weights evolved via STDP while the network learned and tracked spatiotemporal patterns (Figures 10c and 10d). These demonstrations validate that device-level plasticity can be harnessed for autonomous adaptation and unsupervised feature learning in hardware.

Beyond these core demonstrations, several research lines broaden the STDP rule. Studies of memristive stochasticity argue that device randomness can be used as a computational resource rather than only a defect; stochastic updates can improve exploration and generalization in some learning tasks [150]. System and architecture studies emphasize multi-memristor synapses, selector/1T integration, and pulse-engineering as practical routes to scalability and reliability [154,157].

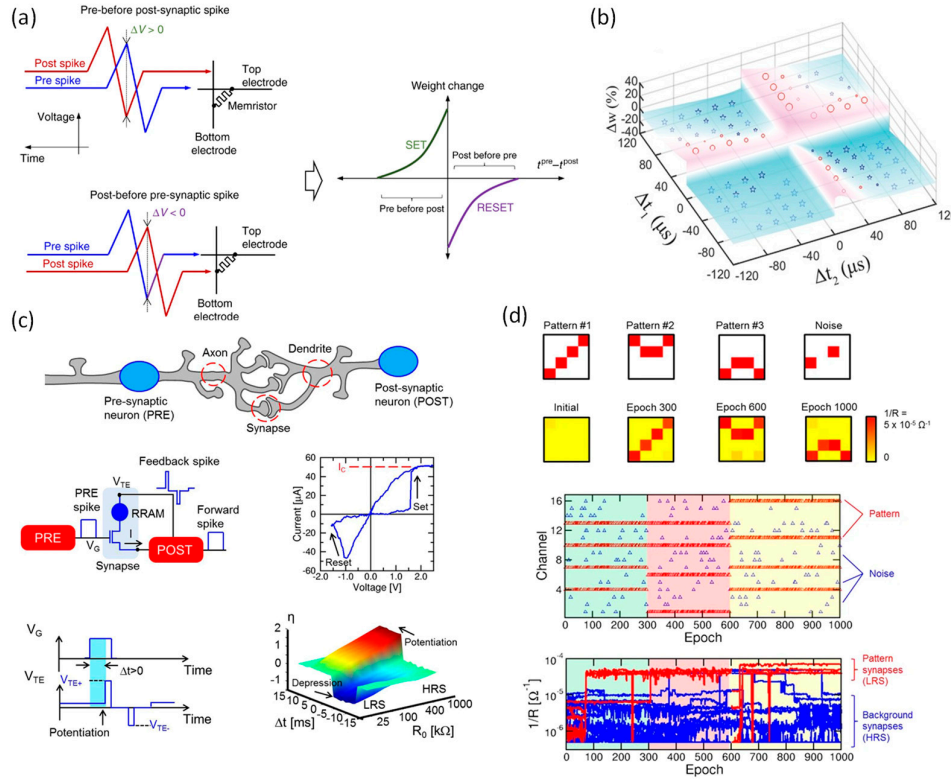


Figure 10. (a) Experimental STDP implementation and measured pair-STDP window: typical pre/post pulse shapes and resulting Δw versus Δt . Reproduced with permission of Ref. [160], Copyright [2018] Nature Communications. (b) Triplet-STDP: weight change under three-spike protocols and comparison with the triplet-STDP model. Reproduced with permission of Ref. [161], Copyright [2017] Advanced Functional Materials. (c) Hardware synapse and neuron module: circuit diagram of synapse interfaced with an LIF neuron. (d) Online pattern learning and tracking: evolution of synaptic weights and system performance on pattern-recognition/tracking tasks. Reproduced with permission of Ref. [162], Copyright [2018] Nature Scientific Reports.

4.5. Memristor-Enabled Stochastic Computation

Memristors exhibit intrinsic, pulse-dependent stochasticity originating from ion migration, filament nucleation/rupture and phase-transition kinetics [163]. Two measurable statistics summarize this behavior. One is the discrete switching probability $p_{\text{switch}}(V, t)$, which captures the chance that a device will transition between states under a given pulse [164–168]. The other is the post-pulse analog conductance distribution, describing the continuous range of conductance values that a device may assume after modulation [169–175].

This device-level randomness can be repurposed as a computational resource by two complementary approaches. The first primitive is probabilistic bits (p-bits). Under near-threshold excitation a memristor produces a binary outcome whose expectation depends sigmoidally on the applied bias; by controlling the bias one therefore controls the sampling probability. Networks of p-bits realize hardware-native stochastic logic and can implement probabilistic primitives with minimal digital overhead. The second primitive is continuous conductance noise for Bayesian sampling and training. A single write pulse may produce a random conductance change; the distribution of such changes can sometimes be approximated as continuous and, under aggregation of multiple independent device currents, may approach a Gaussian perturbation by the central-limit effect. Such perturbations serve as the physical Langevin noise in memristor-based stochastic gradient Langevin dynamics (mSGLD) and other in-memory Bayesian update schemes.

Two representative experimental studies illustrate these modes. $\text{Cu}_{0.1}\text{Te}_{0.9}/\text{HfO}_2/\text{Pt}$ diffusive memristors were demonstrated as reliable p-bit sources, with threshold-switching statistics reported and pulse amplitude mapped to sampling probability; small p-bit networks are used to realize stochastic logic (Figures 11a and 11b) [164]. An in-memory deep Bayesian active-learning system was implemented [170]. This system characterizes post-pulse conductance histograms, uses device noise as the Langevin perturbation in mSGLD, and reports improvements in labeling efficiency and system energy for active learning tasks (Figures 11c–e).

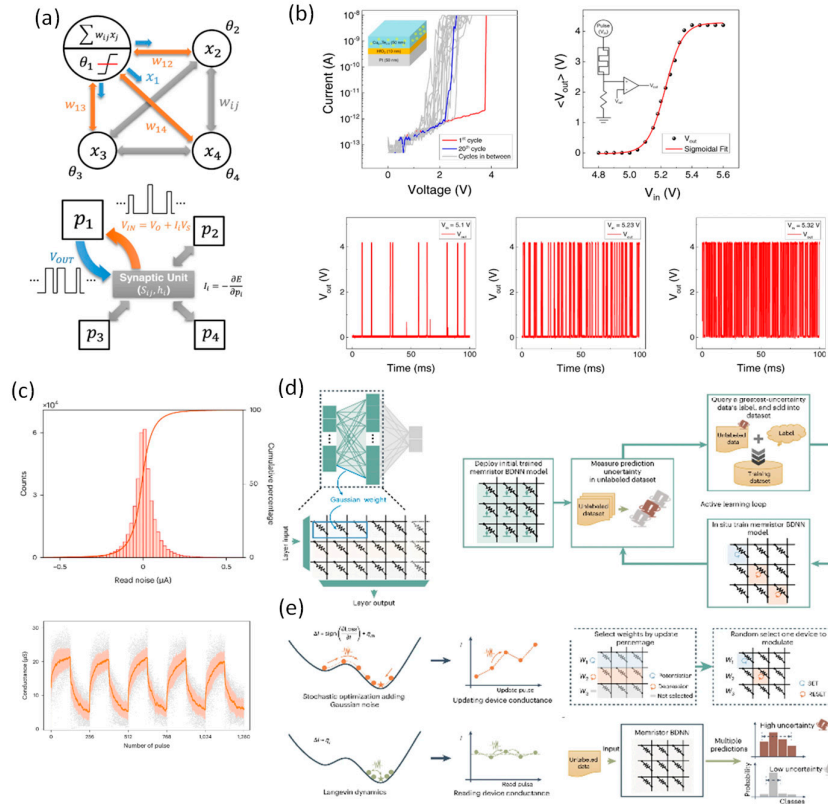


Figure 11. (a–b) Device threshold-switching statistics and p-bit logic mapping: how pulse amplitude controls output probability and how small p-bit networks implement Boolean/probabilistic operations. Reproduced with permission of Ref. [164], Copyright [2022] Nature Communications. (c–e) Conductance-change histograms, the mSGLD in-memory update loop, and system-level active-learning gains linking device-level noise models to Bayesian training and application metrics. Reproduced with permission of Ref. [170], Copyright [2024] Nature Computing Science.

Additional works extend and validate these concepts across device types and applications. Versatile stochastic dot-product circuits were demonstrated that exploit controlled device and circuit variability to implement approximate probabilistic MVM operations suitable for edge inference [169]. Threshold-switching memristors have been used to build stochastic neurons, enabling probabilistic spiking and Boltzmann-style sampling in neuromorphic circuits. NbO_x metal-insulator transition devices have been reported as self-oscillatory p-bits, broadening the set of physical primitives for probabilistic computing [166].

5. Discussion

Memristor-based accelerators for AI—whether leveraging static analog conductances for dense linear algebra or dynamic resistive switching for temporal processing—share a common hardware foundation and a common set of constraints. In both cases, conductance-programming fidelity and array-level nonidealities fundamentally limit achievable accuracy and scalability.

Key device parameters determine suitability for specific computational primitives. For clarity, two operational modes are distinguished (Table 1). Nonvolatile memristors (e.g., VCM, ECM, PCM-like variants) provide persistent analog conductances and are the preferred substrate for dense MVM and many stateful/deterministic functions (MVM, AMC solvers, stateful logic, attractor networks). Volatile memristors (WOx, Ag/Cu diffusive, threshold-switch variants) exhibit fast decay and spontaneous reset, making them well suited for reservoir computing, many spike-timing-dependent plasticity (STDP) implementations, and probabilistic-bit (p-bit) sampling. Stochastic computing generally follows two routes: (a) continuous noisy weights realized with programmed nonvolatile memristors, or (b) p-bit streams generated by volatile threshold or diffusive devices. Mapping any AI primitive onto hardware therefore begins with selecting device physics that match its functional role—persistence versus fading memory, and deterministic versus stochastic behavior.

Table 1. Application statistics of static conductance and dynamic switching.

Application	Typical device	Key feature	Conductance states	Randomness role	Endurance stress	Peripheral criticality
MVM	Nonvolatile memristor, long-term retention	Highly parallel dot-products	Multilevel or analog	Harmful	Low (inference), high (training)	High (DACs/ADCs, drivers)
Analog matrix-equation solving		Closed-loop for matrix inversion/pseudoinversion			High (second-order training), low (other tasks)	High (DACs/ADCs, drivers, op-amp)
Stateful logic		Deterministic SET/RESET, binary	Binary or discretized multilevel		High (frequency switching)	Medium (pulse drivers, selectors)
Attractor networks						
Stochastic computing — continuous noisy weights		analog statistical conductance	Multilevel	Beneficial		Medium (sampling circuitry)
Reservoir computing &	Volatile memristors, short-	Fading memory, nonlinear	Binary or discretized	Beneficial		Medium (analog readout)

spatiotemporal detection	term retention	I–V, tunable τ	multilevel		
STDP		Pulse-induced incremental updates	Analog weight changes	Neutral / mildly beneficial	High (pulse timing, neuron circuits)
Stochastic computing – p-bit		Probabilistic switching	Binary	Beneficial	Low (sampling circuitry)

Important device metrics include: conductance uniformity (device-to-device consistency), intrinsic stochasticity (cycle-to-cycle variability), conductance range, number of distinct conductance states, programming precision (effective bits), switching voltage and consistency, switching speed or relaxation time, endurance (cycle lifetime), and retention. For example, crossbar-based MVM implementations typically achieves effective precisions of only a few bits—around 5–8 bits in practice for task-specific deployments—necessitating programming and mapping methods that meet precision targets [34].

At the array level, parasitic line resistance, sneak paths, and capacitive coupling distort the intended Ohm/Kirchhoff mappings. IR drops limit array size and skew the effective conductances seen at the periphery [176]. Passive 1T1R arrays are especially susceptible; selector integration, array partitioning, and hierarchical tiling are common mitigation strategies [3]. Peripheral circuits are equally critical: high-throughput MVM requires DACs for row-voltage generation and ADCs (or low-overhead analog integrators/transimpedance amplifiers) for column readout [177]. AMC solvers add further analog constraints, embedding arrays in amplifier feedback loops where bandwidth, offset, and stability become primary design concerns [78]. In dynamic applications, pulse drivers, precise timing generators and neuron interfaces are essential.

Device switching speed and relaxation time constants determine both computational throughput and feasible temporal roles. For reservoir computing or STDP, volatile-device time constants (τ) must align with task memory horizons or spike timing windows [133]. Nonvolatile devices used in online training, stateful logic, or attractor networks experience repeated writing, resulting in endurance stress [178]. Even volatile devices, though self-resetting, can degrade under high-frequency pulsing or sustained bias [179]. Endurance requirements therefore depend on both device physics and usage patterns: MVM during inference imposes low stress, whereas on-chip training or analog matrix-equation solving demands high endurance during iterative updates. Dynamic-switching applications, with inherently frequent state changes, generally impose the most stringent endurance constraints. Practical systems must budget write cycles, minimize redundant updates, and employ endurance-aware strategies such as write reduction, wear leveling, or hybrid retention.

Device variability and stochastic switching are double-edged. Uncontrolled variability undermines deterministic operations such as stateful logic or associative recall, yet controlled variability can enrich reservoir dynamics and provide intrinsic randomness for probabilistic inference [5]. Algorithmically, two strategies emerge: (1) suppress variability where determinism is critical through device screening, redundancy, or digital correction; and (2) exploit variability where randomness is beneficial through noise-aware training or stochastic sampling. When high precision is required, bit-parallel or bit-serial mappings are used, albeit with increased sequencing overhead and endurance cost [29].

Mapping large models onto finite arrays involves tiling, batching, and scheduling [42]. Architectures with reusable weights, such as CNNs, benefit from repeated matrix access, which amortizes DAC/ADC overhead and mitigates endurance stress. In contrast, models with frequent

weight updates—such as online-trained RNNs or dynamically updated Transformer key-value caches—impose higher write demands and tighter timing constraints. GNNs and other sparse workloads exhibit irregular access patterns, often inefficient on dense crossbars, motivating hybrid sparse-dense or time-multiplexed designs. Bit-serial quantization can reduce per-device precision requirements but increases control complexity and latency [42,180]. Across the system stack, peripheral power and area often dominate: ADC/DAC resolution, amplifier linearity, and selector/driver overhead all contribute to overall trade-offs [29].

Effective deployment requires co-design across device, circuit, and algorithm levels. At the device-circuit interface, calibrated multi-step programming, verify-and-tune procedures, and selector integration remain essential to mitigate nonlinearity, variability, and IR-drop [29]. Dynamic-switching functionalities introduce additional challenges in timing, retention, reproducibility, and pulse fidelity [181]. For example, stateful logic depends on precise pulse timing and controlled SET/RESET probabilities [114]. Attractor networks require drift stability and tuned feedback gain [129]. Stochastic computing demands well-characterized switching-probability curves, low autocorrelation, and controlled bias [164]. STDP circuits need calibrated spike-pair protocols and symmetric, high-resolution updates [151]. Reservoir processors rely on matched time constants and appropriate masking or multiplexing [136]. At the array level, partitioning, local pre-compensation, and redundancy mitigate distortions from heterogeneity and IR-drop [42]. Architecturally, hybrid approaches are effective—using dense analog MVM for bulk linear algebra, paired with stochastic or temporal modules for task-specific dynamics [134]. Algorithmic remedies such as hardware-in-the-loop training, noise-aware optimization, regularization and preconditioning relax device constraints and improve end-to-end accuracy [75,82]. System-level strategies including wear leveling, periodic calibration, and mixed-precision accumulation help balance endurance, energy, and fidelity.

Static conductance and dynamic switching modalities are complementary and often interdependent. Static conductance enables stable, high-throughput analog computation, while dynamic switching extends functionality to stateful logic and temporal processing. Both rely on the same physical memristor substrate and thus share constraints of variability, endurance, and peripheral overhead—necessitating unified co-design. Ultimately, scalable memristive AI accelerators will depend on architectures that integrate both modalities seamlessly, assign each to its optimal computational role, and coordinate design across materials, circuits, and algorithms to balance precision, adaptability, and reliability.

6. Conclusions

Memristors offer two complementary hardware primitives—static conductance and dynamic switching—that together open new pathways for energy-efficient, low-latency on-device AI. Static conductance in dense crossbar arrays enables highly parallel matrix-vector multiplication and supports both digital in-memory arithmetic and closed-loop AMC topologies. These capabilities directly accelerate core linear-algebra workloads in modern networks, from fully connected and convolutional layers to transformers. Dynamic, volatile memristors provide fading memory and intrinsic nonlinearity. They enable temporal processing primitives such as reservoir computing, spiking networks with STDP, attractor dynamics, and hardware-native stochastic computation. Together, these device modalities permit compact implementations of both data-parallel and time-domain AI primitives.

Significant progress has been made at device, circuit, and system levels. Experimental demonstrations show viable MAC acceleration, one-step analog solvers, reservoir-based speech and biosignal processing, and on-line STDP learning in integrated arrays. At the same time, non-idealities remain the major barrier to deployment. Device variability, stochastic switching, endurance and retention limits, wire parasitics, and peripheral overhead constrain achievable accuracy, throughput, and energy efficiency.

Realizing practical memristor-based AI requires integrated solutions. We advocate coordinated device-circuit-algorithm co-design, hybrid analog-digital architectures that place each primitive

where it is most effective, robust pulse and calibration strategies, and compiler toolchains for mapping algorithms to hardware. Standardized benchmarks and reproducible system demonstrations will be essential to quantify benefits. With these efforts, memristor technologies can move from compelling laboratory prototypes to scalable, application-ready accelerators that bring powerful AI capabilities to edge and embedded platforms.

Author Contributions: Conceptualization, Z.M. and Z.S.; methodology, Z.M.; formal analysis, Z.M. and Z.S.; investigation, Z.M.; resources, Z.M.; writing—original draft preparation, Z.M., S. Z., C. H., Y. L., Y. L. and S. W.; writing—review and editing, Z.M. and Z.S.; visualization, Z.M. and Z.S.; supervision, Z.S.; project administration, Z.S.; funding acquisition, Z.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grant No. 62572011, the Beijing Natural Science Foundation under Grant No. 4252016, and the 111 Project under Grant No. B18001.

Data Availability Statement: No new data were created or analyzed in this study.

Conflicts of Interest: The authors declare no conflicts of interest.

References

- Horowitz, M. Computing's energy problem (and what we can do about it). *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC)* **2014**, pp. 10–14.
- Waldrop, M.M. The chips are down for Moore's law. *Nature* **2016**, 530, 144–147.
- Ielmini, D.; Wong, H.-S.P. In-memory computing with resistive switching devices. *Nat. Electron.* **2018**, 1, 333–343.
- Sun, Z.; Kvatinisky, S.; Si, X.; Mehonic, A.; Cai, Y.; Huang, R. A full spectrum of computing-in-memory technologies. *Nat. Electron.* **2023**, 6, 823–835.
- Sebastian, A.; Le Gallo, M.; Khaddam-Aljameh, R.; Eleftheriou, E. Memory devices and applications for in-memory computing. *Nat. Nanotechnol.* **2020**, 15, 529–544.
- Sun, W.; Gao, B.; Chi, M.; Xia, Q.; Yang, J.J.; Qian, H.; Wu, H. Understanding memristive switching via in situ characterization and device modeling. *Nat. Commun.* **2019**, 10, 3453.
- Ielmini, D. Resistive switching memories based on metal oxides: Mechanisms, reliability and scaling. *Semicond. Sci. Technol.* **2016**, 31, 063002.
- Mehonic, A.; Ielmini, D.; Roy, K.; Mutlu, O.; Kvatinisky, S.; Serrano-Gotarredona, T.; Linares-Barranco, B.; Spiga, S.; Savel'ev, S.; Balanov, A.G.; et al. Roadmap to neuromorphic computing with emerging technologies. *APL Mater.* **2024**, 12, 109201.
- Yao, P.; Wu, H.; Gao, B.; Tang, J.; Zhang, Q.; Zhang, W.; Yang, J.J.; Qian, H. Fully hardware-implemented memristor convolutional neural network. *Nature* **2020**, 577, 641–646.
- Sun, Z.; Pedretti, G.; Ambrosi, E.; Bricalli, A.; Wang, W.; Ielmini, D. Solving matrix equations in one step with cross-point resistive arrays. *Proc. Natl. Acad. Sci. U.S.A.* **2019**, 116, 4123–4128.
- Zhang, G.; Qin, J.; Zhang, Y.; Gong, G.; Xiong, Z.Y.; Ma, X.; Lv, Z.; Zhou, Y.; Han, S.T. Functional materials for memristor-based reservoir computing: Dynamics and applications. *Adv. Funct. Mater.* **2023**, 33, 2302929.
- Midya, R.; Wang, Z.; Asapu, S.; Zhang, X.; Rao, M.; Song, W.; Zhuo, Y.; Upadhyay, N.; Xia, Q.; Yang, J.J. Reservoir computing using diffusive memristors. *Adv. Intell. Syst.* **2019**, 1, 1900084.
- Kumar, S.; Wang, X.; Strachan, J.P.; Yang, Y.; Lu, W.D. Dynamical memristors for higher-complexity neuromorphic computing. *Nat. Rev. Mater.* **2022**, 7, 575–591.
- Burr, G.W.; Shelby, R.M.; Sebastian, A.; Kim, S.; Kim, S.; Sidler, S.; Virwani, K.; Ishii, M.; Narayanan, P.; Fumarola, A.; et al. Neuromorphic computing using non-volatile memory. *Adv. Phys. X* **2017**, 2, 89–124.
- Chua, L.O. Memristor—The missing circuit element. *IEEE Trans. Circuit Theory* **1971**, 18, 507–519.
- Strukov, D.B.; Snider, G.S.; Stewart, D.R.; Williams, R.S. The missing memristor found. *Nature* **2008**, 453, 80–83.

17. Kim, H.; Yang, S.J.; Shim, Y.S.; Moon, C.W. A comprehensive review of electrochemical metallization and valence change mechanisms in filamentary resistive switching of halide perovskite-based memory devices. *ACS Appl. Mater. Interfaces* **2025**, *17*, 50122–50141.
18. Kwon, D.H.; Kim, K.M.; Jang, J.H.; Jeon, J.M.; Lee, M.H.; Kim, G.H.; Li, X.S.; Park, G.S.; Lee, B.; Han, S.; et al. Atomic structure of conducting nanofilaments in TiO₂ resistive switching memory. *Nat. Nanotechnol.* **2010**, *5*, 148–153.
19. Sokolov, A.S.; Jeon, Y.R.; Kim, S.; Ku, B.; Lim, D.; Han, H.; Chae, M.G.; Lee, J.; Ha, B.G.; Choi, C. Influence of oxygen vacancies in ALD HfO_{2-x} thin films on non-volatile resistive switching phenomena with a Ti/HfO_{2-x}/Pt structure. *Appl. Surf. Sci.* **2018**, *434*, 822–830.
20. Zhang, Y.; Mao, G.Q.; Zhao, X.; Li, Y.; Zhang, M.; Wu, Z.; Wu, W.; Sun, H.; Guo, Y.; Wang, L.; et al. Evolution of the conductive filament system in HfO₂-based memristors observed by direct atomic-scale imaging. *Nat. Commun.* **2021**, *12*, 7232.
21. Valov, I.; Waser, R.; Jameson, J.R.; Kozicki, M.N. Electrochemical metallization memories—Fundamentals, applications, prospects. *Nanotechnology* **2011**, *22*, 254003.
22. Ye, C.; Wu, J.; He, G.; Zhang, J.; Deng, T.; He, P.; Wang, H. Physical mechanism and performance factors of metal oxide based resistive switching memory: A review. *J. Mater. Sci. Technol.* **2016**, *32*, 1–11.
23. Raoux, S.; Wehnic, W.; Ielmini, D. Phase change materials and their application to nonvolatile memories. *Chem. Rev.* **2010**, *110*, 240–267.
24. Wuttig, M.; Yamada, N. Phase-change materials for rewriteable data storage. *Nat. Mater.* **2007**, *6*, 824–832.
25. Zhao, Z.; Clima, S.; Garbin, D.; Degraeve, R.; Pourtois, G.; Song, Z.; Zhu, M. Chalcogenide ovonic threshold switching selector. *Nano-Micro Lett.* **2024**, *16*, 81.
26. Han, C.Y.; Han, Z.R.; Fang, S.L.; Fan, S.Q.; Yin, J.Q.; Liu, W.H.; Li, X.; Yang, S.Q.; Zhang, G.H.; Wang, X.L.; et al. Characterization and modelling of flexible VO₂ Mott memristor for the artificial spiking warm receptor. *Adv. Mater. Interfaces* **2022**, *9*, 2200394.
27. Ju, D.; Kim, S. Versatile NbO_x-based volatile memristor for artificial intelligent applications. *Adv. Funct. Mater.* **2024**, *34*, 2409436.
28. Prezioso, M.; Merrih-Bayat, F.; Hoskins, B.D.; Adam, G.C.; Likharev, K.K.; Strukov, D.B. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* **2015**, *521*, 61–64.
29. Le Gallo, M.; Sebastian, A.; Mathis, R.; Giefers, H.; Tuma, T.; Bekas, C.; Curioni, A.; Eleftheriou, E. Mixed-precision in-memory computing. *Nat. Electron.* **2018**, *1*, 246–253.
30. Song, W.; Rao, M.; Li, Y.; Li, C.; Zhuo, Y.; Cai, F.; Wu, M.; Yin, W.; Li, Z.; Wei, Q.; et al. Programming memristor arrays with arbitrarily high precision for analog computing. *Science* **2024**, *383*, 903–910.
31. Ni, L.; Wang, Y.; Yu, H.; Yang, W.; Weng, C.; Zhao, J. An energy-efficient matrix multiplication accelerator by distributed in-memory computing on binary RRAM crossbar. *21st Asia South Pacific Design Automation Conf. (ASP-DAC)*, Macao, **2016**; pp. 280–285.
32. Sun, Z.; Huang, R. Time complexity of in-memory matrix-vector multiplication. *IEEE Trans. Circuits Syst. II: Express Briefs* **2021**, *68*, 2785–2789.
33. Wang, S.; Sun, Z. Dual in-memory computing of matrix-vector multiplication for accelerating neural networks. *Device* **2024**, *2*, 100546.
34. Li, C.; Hu, M.; Li, Y.; Jiang, H.; Ge, N.; Montgomery, J.; Zhang, J.; Song, W.; Dávila, N.; Graves, C.E.; et al. Analogue signal and image processing with large memristor crossbars. *Nat. Electron.* **2018**, *1*, 52–59.
35. Cai, F.; Correll, J.M.; Lee, S.H.; Lim, Y.; Bothra, S.; Zhang, Z.; Flynn, M.P.; Lu, W.D. A fully integrated reprogrammable memristor-CMOS system for efficient multiply-accumulate operations. *Nat. Electron.* **2019**, *2*, 290–299.
36. Chen, W.H.; Li, K.X.; Lin, W.Y.; Hsu, K.H.; Li, P.Y.; Yang, C.H.; Xue, C.X.; Yang, E.Y.; Chen, Y.K.; Chang, Y.S.; et al. A 65nm 1Mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors. *IEEE Int. Solid-State Circuits Conf. (ISSCC)*, **2018**; pp. 494–496.
37. Gokmen, T.; Vlasov, Y. Acceleration of deep neural network training with resistive cross-point devices: Design considerations. *Front. Neurosci.* **2016**, *10*, 333.

38. Wen, T.H.; Hung, J.M.; Huang, W.H.; Jhang, C.J.; Lo, Y.C.; Hsu, H.H.; Ke, Z.E.; Chen, Y.C.; Chin, Y.H.; Su, C.I.; et al. Fusion of memristor and digital compute-in-memory processing for energy-efficient edge computing. *Science* **2024**, *384*, 325–332.
39. Jeon, K.; Ryu, J.J.; Im, S.; Seo, H.K.; Eom, T.; Ju, T.; Yang, M.K.; Jeong, D.S.; Kim, G.H. Purely self-rectifying memristor-based passive crossbar array for artificial neural network accelerators. *Nat. Commun.* **2024**, *15*, 129.
40. Hu, M.; Strachan, J.P.; Li, Z.; Grafals, E.; Davila, N.; Graves, C.; Lam, S.; Ge, N.; Yang, J.J.; Williams, R.S. Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication. *Proc. 53rd Annu. Design Autom. Conf. (DAC)*, **2016**; pp. 1–6.
41. Lecun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444.
42. Wan, W.; Kubendran, R.; Schaefer, C.; Eryilmaz, S.B.; Zhang, W.; Wu, D.; Deiss, S.; Raina, P.; Qian, H.; Gao, B.; et al. A compute-in-memory chip based on resistive random-access memory. *Nature* **2022**, *608*, 504–512.
43. Wen, S.; Wei, H.; Zeng, Z.; Huang, T. Memristive fully convolutional network: An accurate hardware image-segmentor in deep learning. *IEEE Trans. Emerg. Top. Comput. Intell.* **2018**, *2*, 324–334.
44. Hu, M.; Graves, C.E.; Li, C.; Li, Y.; Ge, N.; Montgomery, J.; Davila, N.; Jiang, H.; Williams, R.S.; Yang, J.J.; et al. Memristor-based analog computation and neural network classification with a dot product engine. *Adv. Mater.* **2018**, *30*, 1705914.
45. Yakopcic, C.; Alom, M.Z.; Taha, T.M. Memristor crossbar deep network implementation based on a convolutional neural network. *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, **2016**; pp. 963–970.
46. Migliato Marega, G.; Ji, H.G.; Wang, Z.; Pasquale, G.; Tripathi, M.; Radenovic, A.; Kis, A. A large-scale integrated vector-matrix multiplication processor based on monolayer molybdenum disulfide memories. *Nat. Electron.* **2023**, *6*, 991–998.
47. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780.
48. Li, C.; Wang, Z.; Rao, M.; Belkin, D.; Song, W.; Jiang, H.; Yan, P.; Li, Y.; Lin, P.; Hu, M.; et al. Long short-term memory networks in memristor crossbar arrays. *Nat. Mach. Intell.* **2019**, *1*, 49–57.
49. Wang, Z.; Li, C.; Lin, P.; Rao, M.; Nie, Y.; Song, W.; Qiu, Q.; Li, Y.; Yan, P.; Strachan, J.P.; et al. In situ training of feed-forward and recurrent convolutional memristor networks. *Nat. Mach. Intell.* **2019**, *1*, 434–442.
50. Liu, X.; Zeng, Z.; Wunsch II, D.C. Memristor-based LSTM network with in situ training and its applications. *Neural Netw.* **2020**, *131*, 300–311.
51. Smagulova, K.; James, A.P. A survey on LSTM memristive neural network architectures and applications. *Eur. Phys. J. Spec. Top.* **2019**, *228*, 2313–2324.
52. Nikam, H.; Satyam, S.; Sahay, S. Long short-term memory implementation exploiting passive RRAM crossbar array. *IEEE Trans. Electron Devices* **2021**, *69*, 1743–1751.
53. Koller, D.; Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*; MIT Press: Cambridge, MA, USA, **2009**.
54. Arka, A.I.; Doppa, J.R.; Pande, P.P.; Joardar, B.K.; Chakrabarty, K. ReGraphX: NoC-enabled 3D heterogeneous ReRAM architecture for training graph neural networks. *Design, Automation & Test in Europe Conf. (DATE)*, **2021**; pp. 1667–1672.
55. Lyu, B.; Wang, S.; Wen, S.; Shi, K.; Yang, Y.; Zeng, L.; Huang, T. AutoGMap: Learning to map large-scale sparse graphs on memristive crossbars. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *35*, 12888–12898.
56. Morsali, M.; Nazzal, M.; Khreishah, A.; Angizi, S. Ima-GNN: In-memory acceleration of centralized and decentralized graph neural networks at the edge. *Proc. Great Lakes Symp. VLSI*, **2023**; pp. 3–8.
57. Liu, C.; Liu, H.; Jin, H.; Liao, X.; Zhang, Y.; Duan, Z.; Xu, J.; Li, H. ReGNN: A ReRAM-based heterogeneous architecture for general graph neural networks. *Proc. 59th ACM/IEEE Design Autom. Conf. (DAC)*, **2022**; pp. 469–474.
58. He, Y.; Li, B.; Wang, Y.; Liu, C.; Li, H.; Li, X. A task-adaptive in-situ ReRAM computing for graph convolutional networks. *IEEE Trans. Computer-Aided Des. Integr. Circuits Syst.* **2024**, *43*, 2635–2646.
59. Ogbogu, C.O.; Arka, A.I.; Pfromm, L.; Joardar, B.K.; Doppa, J.R.; Chakrabarty, K.; Pande, P.P. Accelerating graph neural network training on ReRAM-based PIM architectures via graph and model pruning. *IEEE Trans. Computer-Aided Des. Integr. Circuits Syst.* **2022**, *42*, 2703–2716.

60. Yu, Y.; Wang, S.; Xu, M.; Wen, S.; Zhang, W.; Wang, B.; Yang, J.; Wang, S.; Zhang, Y.; Wu, X.; et al. Random memristor-based dynamic graph CNN for efficient point cloud learning at the edge. *npj Unconventional Comput.* **2024**, *1*, 6.
61. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
62. Yang, X.; Yan, B.; Li, H.; Chen, Y. ReTransformer: ReRAM-based processing-in-memory architecture for transformer acceleration. *Proc. 39th Int. Conf. Computer-Aided Design (ICCAD)*, **2020**; pp. 1–9.
63. Sridharan, S.; Stevens, J.R.; Roy, K.; Raghunathan, A. X-former: In-memory acceleration of transformers. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2023**, *31*, 1223–1233.
64. Leroux, N.; Manea, P.P.; Sudarshan, C.; Finkbeiner, J.; Siegel, S.; Strachan, J.P.; Neftci, E. Analog in-memory computing attention mechanism for fast and energy-efficient large language models. *Nat. Comput. Sci.* **2025**, *5*, 813–824.
65. Park, J.; Choi, J.; Kyung, K.; Kim, M.J.; Kwon, Y.; Kim, N.S.; Ahn, J.H. Attacc! Unleashing the power of PIM for batched transformer-based generative model inference. *Proc. 29th ACM Int. Conf. Architectural Support Program. Lang. Operating Syst. (ASPLOS)*, **2024**; pp. 103–119.
66. Zhou, M.; Xu, W.; Kang, J.; Rosing, T. TransPIM: A memory-based acceleration via software-hardware co-design for transformer. *Proc. IEEE Int. Symp. High-Performance Computer Architecture (HPCA)*, **2022**; pp. 1071–1085.
67. Bettayeb, M.; Halawani, Y.; Khan, M.U.; Saleh, H.; Mohammad, B. Efficient memristor accelerator for transformer self-attention functionality. *Sci. Rep.* **2024**, *14*, 24173.
68. Yang, C.; Wang, X.; Zeng, Z. Full-circuit implementation of transformer network based on memristor. *IEEE Trans. Circuits Syst. I: Regular Papers* **2022**, *69*, 1395–1407.
69. Duan, Q.; Jing, Z.; Zou, X.; Wang, Y.; Yang, K.; Zhang, T.; Wu, S.; Huang, R.; Yang, Y. Spiking neurons with spatiotemporal dynamics and gain modulation for monolithically integrated memristive neural networks. *Nat. Commun.* **2020**, *11*, 3399.
70. Nowshin, F.; Yi, Y. Memristor-based deep spiking neural network with a computing-in-memory architecture. *Proc. 23rd Int. Symp. Quality Electron. Design (ISQED)*, **2022**; pp. 1–6.
71. Sun, Z.; Ielmini, D. Invited tutorial: Analog matrix computing with crosspoint resistive memory arrays. *IEEE Trans. Circuits Syst. II: Express Briefs* **2022**, *69*, 3024–3029.
72. Sun, Z.; Pedretti, G.; Mannocci, P.; Ambrosi, E.; Bricalli, A.; Ielmini, D. Time complexity of in-memory solution of linear systems. *IEEE Trans. Electron Devices* **2020**, *67*, 2945–2951.
73. Sun, Z.; Pedretti, G.; Ambrosi, E.; Bricalli, A.; Ielmini, D. In-Memory Eigenvector Computation in Time $O(1)$. *Adv. Intell. Syst.* **2020**, *2*, 2000042.
74. Sun, Z.; Ambrosi, E.; Pedretti, G.; Bricalli, A.; Ielmini, D. In-memory PageRank accelerator with a cross-point array of resistive memories. *IEEE Trans. Electron Devices* **2020**, *67*, 1466–1470.
75. Hong, C.; Luo, Y.; Zuo, P.; Wang, S.; Sun, Z. Solving All Eigenpairs with Resistive Memory-Based Analog Matrix Computing Circuits. *IEEE Trans. Circuits Syst. I: Reg. Papers* **2025**, early access.
76. Wang, S.; Luo, Y.; Zuo, P.; Pan, L.; Li, Y.; Sun, Z. In-memory analog solution of compressed sensing recovery in one step. *Sci. Adv.* **2023**, *9*, eadj2908.
77. Kutner, M.H.; Nachtsheim, C.J.; Neter, J.; Li, W. *Applied Linear Statistical Models*, 5th ed.; McGraw-Hill: New York, NY, USA, **2004**.
78. Sun, Z.; Pedretti, G.; Bricalli, A.; Ielmini, D. One-step regression and classification with cross-point resistive memory arrays. *Sci. Adv.* **2020**, *6*, eaay2378.
79. Cucchi, M.; Abreu, S.; Ciccone, G.; Brunner, D.; Kleemann, H. Hands-on reservoir computing: A tutorial for practical implementation. *Neuromorph. Comput. Eng.* **2022**, *2*, 032002.
80. Farronato, M.; Mannocci, P.; Melegari, M.; Ricci, S.; Compagnoni, C.M.; Ielmini, D. Reservoir computing with charge-trap memory based on a MoS_2 channel for neuromorphic engineering. *Adv. Mater.* **2023**, *35*, 2205381.
81. Mikolajick, T.; Park, M.H.; Begon-Lours, L.; Slesazek, S. From ferroelectric material optimization to neuromorphic devices. *Adv. Mater.* **2023**, *35*, 2206042.

82. Li, H.; Wang, S.; Zhang, X.; Wang, W.; Yang, R.; Sun, Z.; Feng, W.; Lin, P.; Wang, Z.; Sun, L.; et al. Memristive crossbar arrays for storage and computing applications. *Adv. Intell. Syst.* **2021**, *3*, 2100017.
83. Wang, S.; Li, Y.; Wang, D.; Zhang, W.; Chen, X.; Dong, D.; Wang, S.; Zhang, X.; Lin, P.; Gallichio, C. Echo state graph neural networks with analogue random resistive memory arrays. *Nat. Mach. Intell.* **2023**, *5*, 104–113.
84. Mikhaylov, A.; Pimashkin, A.; Pigareva, Y.; Gerasimova, S.; Gryaznov, E.; Shchanikov, S.; Zuev, A.; Talanov, M.; Lavrov, I.; Demin, V. Neurohybrid memristive CMOS-integrated systems for biosensors and neuroprosthetics. *Front. Neurosci.* **2020**, *14*, 358.
85. Gómez-Luna, J.; Guo, Y.; Brocard, S.; Legriel, J.; Cimadomo, R.; Oliveira, G.F. Evaluating machine learning workloads on memory-centric computing systems. *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, **2023**; pp. 35–49.
86. Wang, S.; Sun, Z.; Liu, Y.; Bao, S.; Cai, Y.; Ielmini, D.; Huang, R. Optimization Schemes for In-Memory Linear Algebra Using Memristors. *IEEE Trans. Circuits Syst. I: Reg. Papers* **2021**, *68*, 4900–4909.
87. Mannocci, P.; Pedretti, G.; Giannone, E.; Melacarne, E.; Sun, Z.; Ielmini, D. A universal, analog, in-memory computing primitive for linear algebra using memristors. *IEEE Trans. Circuits Syst. I: Reg. Papers* **2021**, *68*, 4889–4899.
88. Mannocci, P.; Melacarne, E.; Ielmini, D. An Analogue In-Memory Ridge Regression Circuit with Application to Massive MIMO Acceleration. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2022**, *12*, 952–962.
89. Mannocci, P.; Ielmini, D. A Generalized Block-Matrix Circuit for Closed-Loop Analog In-Memory Computing. *IEEE J. Explor. Solid-State Comput. Devices Circuits* **2023**, *9*, 47–58.
90. Yao, Z.; Gholami, A.; Shen, S.; Mustafa, M.; Keutzer, K.; Mahoney, M. ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 10665–10673.
91. Zhao, Y.; Jiang, L.; Gao, M.; Jing, N.; Gu, C.; Tang, Q.; Liu, F.; Yang, T.; Liang, X. RePAST: a ReRAM-based PIM accelerator for second-order training of DNN. *arXiv preprint* **2022**, arXiv:2210.15255.
92. Melanson, D.; Abu Khater, M.; Aifer, M.; Donatella, K.; Hunter Gordon, M.; Ahle, T.; Crooks, G.; Martinez, A.J.; Sbahi, F.; Coles, P.J. Thermodynamic Computing System for AI Applications. *Nat. Commun.* **2025**, *16*, 3757.
93. Aifer, M.; Donatella, K.; Gordon, M.H.; Duffield, S.; Ahle, T.; Simpson, D.; Crooks, G.; Coles, P.J. Thermodynamic linear algebra. *npj Unconventional Computing* **2024**, *1*, 13.
94. Coles, P.J.; Aifer, M.; Donatella, K.; Melanson, D.; Gordon, M.H.; Ahle, T.D.; Simpson, D.; Crooks, G.; Martinez, A.J.; Sbahi, F.M. Thermodynamic AI and thermodynamic linear algebra. *Machine Learning with New Compute Paradigms* **2023**, 21.
95. Aifer, M.; Duffield, S.; Donatella, K.; Melanson, D.; Klett, P.; Belateche, Z.; Crooks, G.; Martinez, A.J.; Coles, P.J. Thermodynamic bayesian inference. *Proc. IEEE Int. Conf. Rebooting Comput. (ICRC)* **2024**, pp. 1–13.
96. Donatella, K.; Duffield, S.; Aifer, M.; Melanson, D.; Crooks, G.; Coles, P.J. Thermodynamic natural gradient descent. *npj Unconventional Computing* **2026**, *3*, 5.
97. Donatella, K.; Duffield, S.; Melanson, D.; Aifer, M.; Klett, P.; Salegame, R.; Belateche, Z.; Crooks, G.; Martinez, A.J.; Coles, P.J. Scalable Thermodynamic Second-order Optimization. *arXiv preprint* **2025**, arXiv:2502.08603.
98. Boyd, S.; Vandenberghe, L. *Convex Optimization*; Cambridge University Press: Cambridge, U.K., **2004**.
99. Wu, L.; Braatz, R.D. A Quadratic Programming Algorithm with $O(n^3)$ Time Complexity. *arXiv preprint* **2025**, arXiv:2507.04515.
100. Cohen, M.B.; Lee, Y.T.; Song, Z. Solving Linear Programs in the Current Matrix Multiplication Time. *J. ACM* **2021**, *68*, 1–39.
101. Liu, S.; Wang, Y.; Fardad, M.; Varshney, P.K. A Memristor-Based Optimization Framework for Artificial Intelligence Applications. *IEEE Circuits Syst. Mag.* **2018**, *18*, 29–44.
102. Shang, L.; Adil, M.; Madani, R.; Pan, C. Memristor-Based Analog Recursive Computation Circuit for Linear Programming Optimization. *IEEE J. Explor. Solid-State Comput. Devices Circuits* **2020**, *6*, 53–61.
103. Gao, X.; Liao, L.Z. A New One-Layer Neural Network for Linear and Quadratic Programming. *IEEE Trans. Neural Netw.* **2010**, *21*, 918–929.

104. Di Marco, M.; Forti, M.; Pancioni, L.; Innocenti, G.; Tesi, A. Memristor Neural Networks for Linear and Quadratic Programming Problems. *IEEE Trans. Cybern.* **2022**, *52*, 1822–1835.
105. Liu, Z.; Cheng, H.C.; Hossain, D.; Meng, R.; Bena, Y.; Shi, B.; Chen, D.W.; Yang, S.; Su, Y.; Wang, P.; et al. Ultrafast Hybrid Computing Systems Enabled by Memristor-Based Quadratic Programming Circuits. *Adv. Funct. Mater.* **2024**, *34*, 2401600.
106. Vichik, S.; Borrelli, F. Solving linear and quadratic programs with an analog circuit. *Comput. Chem. Eng.* **2014**, *70*, 160–171.
107. Vichik, S.; Arcak, M.; Borrelli, F. Stability of an analog optimization circuit for quadratic programming. *Syst. Control Lett.* **2016**, *88*, 68–74.
108. Mannocci, P.; Giannone, E.; Ielmini, D. In-Memory Principal Component Analysis by Analogue Closed-Loop Eigendecomposition. *IEEE Trans. Circuits Syst. II: Express Briefs* **2024**, *71*, 1839–1843.
109. Von Luxburg, U. A Tutorial on Spectral Clustering. *Stat. Comput.* **2007**, *17*, 395–416.
110. Chi, Y.; Song, X.; Zhou, D.; Hino, K.; Tseng, B.L. Evolutionary Spectral Clustering by Incorporating Temporal Smoothness. *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Disc. Data Mining* **2007**, pp. 153–162.
111. Leskovec, J.; Lang, K.J.; Mahoney, M. Empirical Comparison of Algorithms for Network Community Detection. *Proc. 19th Int. World Wide Web Conf. (WWW)* **2010**, pp. 631–640.
112. Siddique, N.; Paheding, S.; Elkin, C.P.; Devabhaktuni, V. U-Net and Its Variants for Medical Image Segmentation: A Review of Theory and Applications. *IEEE Access* **2021**, *9*, 82031–82057.
113. Borghetti, J.; Snider, G.S.; Kuekes, P.J.; Yang, J.J.; Stewart, D.R.; Williams, R.S. 'Memristive' Switches Enable 'Stateful' Logic Operations via Material Implication. *Nature* **2010**, *464*, 873–876.
114. Kvatinsky, S.; Belousov, D.; Liman, S.; Satat, G.; Wald, N.; Friedman, E.G.; Kolodny, A.; Weiser, U.C. MAGIC—Memristor-Aided Logic. *IEEE Trans. Circuits Syst. II: Express Briefs* **2014**, *61*, 895–899.
115. Sun, Z.; Ambrosi, E.; Bricalli, A.; Ielmini, D. Logic Computing with Stateful Neural Networks of Resistive Switches. *Adv. Mater.* **2018**, *30*, 1802554.
116. Song, Y.; Wang, X.; Wu, Q.; Yang, F.; Wang, C.; Wang, M.; Miao, X. Reconfigurable and Efficient Implementation of 16 Boolean Logics and Full-Adder Functions with Memristor Crossbar for Beyond von Neumann In-Memory Computing. *Adv. Sci.* **2022**, *9*, 2200036.
117. Miao, Z.; Li, Y.; Wang, S.; Sun, Z. Hybrid Quantum and In-Memory Computing for Accelerating Solving Simon's Problem. *APL Mach. Learn.* **2025**, *3*, 026117.
118. Kim, Y.R.; Shin, D.H.; Ghenzi, N.; Cheong, S.; Cho, J.M.; Shim, S.K.; Lee, J.K.; Kim, B.S.; Yim, S.; Park, T.; et al. Fully Resistive In-Memory Cryptographic Engine with Dual-Polarity Memristive Crossbar Array. *Adv. Funct. Mater.* **2025**, *35*, 2414332.
119. Ben-Hur, R.; Ronen, R.; Haj-Ali, A.; Bhattacharjee, D.; Eliahu, A.; Peled, N.; Kvatinsky, S. SIMPLER MAGIC: Synthesis and Mapping of In-Memory Logic Executed in a Single Row to Improve Throughput. *IEEE Trans. Computer-Aided Des. Integr. Circuits Syst.* **2019**, *39*, 2434–2447.
120. Imani, M.; Gupta, S.; Kim, Y.; Rosing, T. FloatPIM: In-Memory Acceleration of Deep Neural Network Training with High Precision. *Proc. Int. Symp. Comput. Arch. (ISCA)* **2019**, *46*, pp. 802–815.
121. Lu, Z.; Wang, X.; Arafat, M.T.; Yang, H.; Liu, Z.; Zhang, J.; Qu, G. An RRAM-Based Computing-in-Memory Architecture and Its Application in Accelerating Transformer Inference. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2023**, *32*, 485–496.
122. Imani, M.; Gupta, S.; Rosing, T. Digital-Based Processing in-Memory: A Highly-Parallel Accelerator for Data Intensive Applications. *Proc. Int. Symp. Memory Syst.* **2019**, pp. 38–40.
123. Leitersdorf, O.; Ronen, R.; Kvatinsky, S. Making Memristive Processing-in-Memory Reliable. *Proc. IEEE Int. Conf. Electron. Circuits Syst. (ICECS)* **2021**, pp. 1–6.
124. Leitersdorf, O.; Ronen, R.; Kvatinsky, S. ConvPIM: Evaluating Digital Processing-in-Memory through Convolutional Neural Network Acceleration. *arXiv preprint* **2025**, arXiv:2305.04122.
125. Lu, Z.; Arafat, M.T.; Qu, G. RIME: A Scalable and Energy-Efficient Processing-in-Memory Architecture for Floating-Point Operations. *Proc. 26th Asia South Pac. Des. Autom. Conf. (ASP-DAC)* **2021**, pp. 120–125.
126. Jin, H.; Liu, C.; Liu, H.; Luo, R.; Xu, J.; Mao, F.; Liao, X. ReHy: A ReRAM-Based Digital/Analog Hybrid PIM Architecture for Accelerating CNN Training. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *33*, 2872–2884.

127. Imani, M.; Pampana, S.; Gupta, S.; Zhou, M.; Kim, Y.; Rosing, T. DUAL: Acceleration of Clustering Algorithms Using Digital-Based Processing in-Memory. *Proc. 53rd IEEE/ACM Int. Symp. Microarchitecture (MICRO)* **2020**, pp. 356–371.
128. Rashed, M.R.H.; Jha, S.K.; Ewetz, R. Hybrid Analog-Digital In-Memory Computing. *Proc. IEEE/ACM Int. Conf. Computer Aided Des. (ICCAD)* **2021**, pp. 1–9.
129. Li, Y.; Wang, S.; Yang, K.; Yang, Y.; Sun, Z. An Emergent Attractor Network in a Passive Resistive Switching Circuit. *Nat. Commun.* **2024**, *15*, 7683.
130. Hu, S.G.; Liu, Y.; Liu, Z.; Chen, T.P.; Wang, J.J.; Yu, Q.; Deng, L.J.; Yin, Y.; Hosaka, S. Associative Memory Realized by a Reconfigurable Memristive Hopfield Neural Network. *Nat. Commun.* **2015**, *6*, 7522.
131. Cai, F.; Kumar, S.; Van Vaerenbergh, T.; Sheng, X.; Liu, R.; Li, C.; Liu, Z.; Foltin, M.; Yu, S.; Xia, Q.; et al. Power-Efficient Combinatorial Optimization Using Intrinsic Noise in Memristor Hopfield Neural Networks. *Nat. Electron.* **2020**, *3*, 409–418.
132. Zhou, Y.; Wu, H.; Gao, B.; Wu, W.; Xi, Y.; Yao, P.; Zhang, S.; Zhang, Q.; Qian, H. Associative Memory for Image Recovery with a High-Performance Memristor Array. *Adv. Funct. Mater.* **2019**, *29*, 1900155.
133. Liang, X.; Tang, J.; Zhong, Y.; Gao, B.; Qian, H.; Wu, H. Physical Reservoir Computing with Emerging Electronics. *Nat. Electron.* **2024**, *7*, 193–206.
134. Jang, Y.H.; Han, J.K.; Hwang, C.S. A Review of Memristive Reservoir Computing for Temporal Data Processing and Sensing. *InfoScience* **2024**, *1*, e12013.
135. Moon, J.; Ma, W.; Shin, J.H.; Cai, F.; Du, C.; Lee, S.H.; Lu, W.D. Temporal Data Classification and Forecasting Using a Memristor-Based Reservoir Computing System. *Nat. Electron.* **2019**, *2*, 480–487.
136. Du, C.; Cai, F.; Zidan, M.A.; Ma, W.; Lee, S.H.; Lu, W.D. Reservoir Computing Using Dynamic Memristors for Temporal Information Processing. *Nat. Commun.* **2017**, *8*, 2204.
137. Zhong, Y.; Tang, J.; Li, X.; Gao, B.; Qian, H.; Wu, H. Dynamic Memristor-Based Reservoir Computing for High-Efficiency Temporal Signal Processing. *Nat. Commun.* **2021**, *12*, 408.
138. Zhong, Y.; Tang, J.; Li, X.; Liang, X.; Liu, Z.; Li, Y.; Xi, Y.; Yao, P.; Hao, Z.; Gao, B.; et al. A Memristor-Based Analogue Reservoir Computing System for Real-Time and Power-Efficient Signal Processing. *Nat. Electron.* **2022**, *5*, 672–681.
139. Zhang, T.; Wozniak, S.; Syed, G.S.; Mannocci, P.; Farronato, M.; Ielmini, D.; Sebastian, A.; Yang, Y. Emerging Materials and Computing Paradigms for Temporal Signal Analysis. *Adv. Mater.* **2025**, *37*, 2408566.
140. Wang, S.; Gao, S.; Tang, C.; Occhipinti, E.; Li, C.; Wang, S.; Wang, J.; Zhao, H.; Hu, G.; Nathan, A.; Dahiya, R.; et al. Memristor-Based Adaptive Neuromorphic Perception in Unstructured Environments. *Nat. Commun.* **2024**, *15*, 4671.
141. Wang, Y.; Gong, Y.; Huang, S.; Xing, X.; Lv, Z.; Wang, J.; Yang, J.Q.; Zhang, G.; Zhou, Y.; Han, S.T. Memristor-Based Biomimetic Compound Eye for Real-Time Collision Detection. *Nat. Commun.* **2021**, *12*, 5979.
142. Jang, Y.H.; Kim, W.; Kim, J.; Woo, K.S.; Lee, H.J.; Jeon, J.W.; Shim, S.K.; Han, J.; Hwang, C.S. Time-Varying Data Processing with Nonvolatile Memristor-Based Temporal Kernel. *Nat. Commun.* **2021**, *12*, 5727.
143. Yoon, R.; Oh, S.; Cho, S.; Min, K.S. Memristor-CMOS Hybrid Circuits Implementing Event-Driven Neural Networks for Dynamic Vision Sensor Camera. *Micromachines* **2024**, *15*, 426.
144. D'Agostino, S.; Moro, F.; Torchet, T.; Demirağ, Y.; Grenouillet, L.; Castellani, N.; Indiveri, G.; Vianello, E.; Payvand, M. DenRAM: Neuromorphic Dendritic Architecture with RRAM for Efficient Temporal Processing with Delays. *Nat. Commun.* **2024**, *15*, 3446.
145. Wu, X.; Dang, B.; Zhang, T.; Wu, X.; Yang, Y. Spatiotemporal Audio Feature Extraction with Dynamic Memristor-Based Time-Surface Neurons. *Sci. Adv.* **2024**, *10*, ead12767.
146. Milozzi, A.; Ricci, S.; Ielmini, D. Memristive Tonotopic Mapping with Volatile Resistive Switching Memory Devices. *Nat. Commun.* **2024**, *15*, 2812.
147. Ji, X.; Paulsen, B.D.; Chik, G.K.K.; Wu, R.; Yin, Y.; Chan, P.K.L.; Rivnay, J. Mimicking Associative Learning Using an Ion-Trapping Non-Volatile Synaptic Organic Electrochemical Transistor. *Nat. Commun.* **2021**, *12*, 2480.
148. Bi, G.Q.; Poo, M.M. Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type. *J. Neurosci.* **1998**, *18*, 10464–10472.

149. Roy, K.; Jaiswal, A.; Panda, P. Towards Spike-Based Machine Intelligence with Neuromorphic Computing. *Nature* **2019**, *575*, 607–617.
150. Ignatov, M.; Ziegler, M.; Hansen, M.; Kohlstedt, H. Memristive Stochastic Plasticity Enables Mimicking of Neural Synchrony: Memristive Circuit Emulates an Optical Illusion. *Sci. Adv.* **2017**, *3*, e1700849.
151. Jo, S.H.; Chang, T.; Ebong, I.; Bhadviya, B.B.; Mazumder, P.; Lu, W. Nanoscale Memristor Device as Synapse in Neuromorphic Systems. *Nano Lett.* **2010**, *10*, 1297–1301.
152. Weilenmann, C.; Ziogas, A.N.; Zellweger, T.; Portner, K.; Mladenović, M.; Kaniselman, M.; Moraitis, T.; Luisier, M.; Emboras, A. Single Neuromorphic Memristor Closely Emulates Multiple Synaptic Mechanisms for Energy Efficient Neural Networks. *Nat. Commun.* **2024**, *15*, 6898.
153. Midya, R.; Pawar, A.S.; Pattnaik, D.P.; Mooshagian, E.; Borisov, P.; Albright, T.D.; Snyder, L.H.; Williams, R.S.; Yang, J.J.; Balanov, A.G.; et al. Artificial Transneurons Emulate Neuronal Activity in Different Areas of Brain Cortex. *Nat. Commun.* **2025**, *16*, 7289.
154. Ambrogio, S.; Balatti, S.; Milo, V.; Carboni, R.; Wang, Z.Q.; Calderoni, A.; Ramaswamy, N.; Ielmini, D. Neuromorphic Learning and Recognition with One-Transistor-One-Resistor Synapses and Bistable Metal Oxide RRAM. *IEEE Trans. Electron Devices* **2016**, *63*, 1508–1515.
155. Indiveri, G.; Linares-Barranco, B.; Legenstein, R.; Deligeorgis, G.; Prodromakis, T. Integration of Nanoscale Memristor Synapses in Neuromorphic Computing Architectures. *Nanotechnology* **2013**, *24*, 384010.
156. Ohno, T.; Hasegawa, T.; Tsuruoka, T.; Terabe, K.; Gimzewski, J.K.; Aono, M. Short-Term Plasticity and Long-Term Potentiation Mimicked in Single Inorganic Synapses. *Nat. Mater.* **2011**, *10*, 591–595.
157. Boybat, I.; Le Gallo, M.; Nandakumar, S.R.; Moraitis, T.; Parnell, T.; Tuma, T.; Rajendran, B.; Leblebici, Y.; Sebastian, A.; Eleftheriou, E. Neuromorphic Computing with Multi-Memristive Synapses. *Nat. Commun.* **2018**, *9*, 2514.
158. Serrano-Gotarredona, T.; Masquelier, T.; Prodromakis, T.; Indiveri, G.; Linares-Barranco, B. STDP and STDP Variations with Memristors for Spiking Neuromorphic Learning Systems. *Front. Neurosci.* **2013**, *7*, 2.
159. Kheradpisheh, S.R.; Ganjtabesh, M.; Thorpe, S.J.; Masquelier, T. STDP-Based Spiking Deep Convolutional Neural Networks for Object Recognition. *Neural Netw.* **2018**, *99*, 56–67.
160. Prezioso, M.; Mahmoodi, M.R.; Merrikh Bayat, F.; Nili, H.; Kim, H.; Vincent, A.; Strukov, D.B. Spike-Timing-Dependent Plasticity Learning of Coincidence Detection with Passively Integrated Memristive Circuits. *Nat. Commun.* **2018**, *9*, 5311.
161. Yang, R.; Huang, H.M.; Hong, Q.H.; Yin, X.B.; Tan, Z.H.; Shi, T.; Zhou, Y.X.; Miao, X.S.; Wang, X.P.; Mi, S.B.; et al. Synaptic Suppression Triplet-STDP Learning Rule Realized in Second-Order Memristors. *Adv. Funct. Mater.* **2018**, *28*, 1704455.
162. Pedretti, G.; Milo, V.; Ambrogio, S.; Carboni, R.; Bianchi, S.; Calderoni, A.; Ramaswamy, N.; Ielmini, D. Memristive Neural Network for On-Line Learning and Tracking with Brain-Inspired Spike Timing Dependent Plasticity. *Sci. Rep.* **2017**, *7*, 5288.
163. Ding, C.; Ren, Y.; Liu, Z.; Wong, N. Transforming Memristor Noises into Computational Innovations. *Commun. Mater.* **2025**, *6*, 149.
164. Woo, K.S.; Kim, J.; Han, J.; Kim, W.; Jang, Y.H.; Hwang, C.S. Probabilistic Computing Using $\text{Cu}_{0.1}\text{Te}_{0.9}/\text{HfO}_2/\text{Pt}$ Diffusive Memristors. *Nat. Commun.* **2022**, *13*, 5762.
165. Wang, K.; Hu, Q.; Gao, B.; Lin, Q.; Zhuge, F.; Zhang, D.; Wang, L.; He, Y.; Scheicher, R.H.; Tong, H.; Miao, X. Threshold Switching Memristor-Based Stochastic Neurons for Probabilistic Computing. *Mater. Horiz.* **2021**, *8*, 619–629.
166. Rhee, H.; Kim, G.; Song, H.; Park, W.; Kim, D.H.; In, J.H.; Lee, Y.; Kim, K.M. Probabilistic Computing with NbO_x Metal-Insulator Transition-Based Self-Oscillatory pbit. *Nat. Commun.* **2023**, *14*, 7199.
167. Song, L.; Liu, P.; Pei, J.; Liu, Y.; Liu, S.; Wang, S.; Ng, L.W.T.; Hasan, T.; Pun, K.; Gao, S. Lightweight Error-Tolerant Edge Detection Using Memristor-Enabled Stochastic Computing. *Nat. Commun.* **2025**, *16*, 4550.
168. Lin, Y.; Gao, B.; Tang, J.; Zhang, Q.; Qian, H.; Wu, H. Deep Bayesian Active Learning Using In-Memory Computing Hardware. *Nat. Comput. Sci.* **2025**, *5*, 27–36.
169. Mahmoodi, M.R.; Prezioso, M.; Strukov, D.B. Versatile Stochastic Dot Product Circuits Based on Nonvolatile Memories for High Performance Neurocomputing and Neurooptimization. *Nat. Commun.* **2019**, *10*, 5113.

170. Lin, Y.; Zhang, Q.; Gao, B.; Tang, J.; Yao, P.; Li, C.; Huang, S.; Liu, Z.; Zhou, Y.; Liu, Y.; et al. Uncertainty Quantification via a Memristor Bayesian Deep Neural Network for Risk-Sensitive Reinforcement Learning. *Nat. Mach. Intell.* **2023**, *5*, 714–723.
171. Li, X.; Gao, B.; Qin, Q.; Yao, P.; Li, J.; Zhao, H.; Liu, C.; Zhang, Q.; Hao, Z.; Li, Y.; et al. Federated Learning Using a Memristor Compute-in-Memory Chip with In Situ Physical Unclonable Function and True Random Number Generator. *Nat. Electron.* **2025**, *8*, 518–528.
172. Gao, D.; Huang, Q.; Zhang, G.L.; Yin, X.; Li, B.; Schlichtmann, U.; Zhuo, C. Bayesian Inference Based Robust Computing on Memristor Crossbar. *Proc. 58th Des. Autom. Conf. (DAC)* **2021**, pp. 121–126.
173. Lin, Y.; Zhang, Q.; Tang, J.; Gao, B.; Li, C.; Yao, P.; Liu, Z.; Zhu, J.; Lu, J.; Hu, X.S.; et al. Bayesian Neural Network Realization by Exploiting Inherent Stochastic Characteristics of Analog RRAM. *Proc. IEEE Int. Electron Devices Meet. (IEDM)* **2019**, pp. 14.6.1–14.6.4.
174. Bonnet, D.; Hirtzlin, T.; Majumdar, A.; Dalgaty, T.; Esmahotto, E.; Meli, V.; Castellani, N.; Martin, S.; Nodin, J.F.; Bourgeois, G.; et al. Bringing Uncertainty Quantification to the Extreme-Edge with Memristor-Based Bayesian Neural Networks. *Nat. Commun.* **2023**, *14*, 7530.
175. Gao, D.; Yang, Z.; Huang, Q.; Zhang, G.L.; Yin, X.; Li, B.; Schlichtmann, U.; Zhuo, C. BRoCoM: A Bayesian Framework for Robust Computing on Memristor Crossbar. *IEEE Trans. Computer-Aided Des. Integr. Circuits Syst.* **2022**, *42*, 2136–2148.
176. Shi, L.; Zheng, G.; Tian, B.; Dkhil, B.; Duan, C. Research Progress on Solutions to the Sneak Path Issue in Memristor Crossbar Arrays. *Nanoscale Adv.* **2020**, *2*, 1811–1827.
177. Zolfagharijad, M.; Büchel, J.; Cassola, L.; Kinge, S.; Syed, G.S.; Sebastian, A.; van der Wiel, W.G. Analogue Speech Recognition Based on Physical Computing. *Nature* **2025**, *645*, 886–892.
178. Lanza, M.; Waser, R.; Ielmini, D.; Yang, J.J.; Goux, L.; Suñé, J.; Kenyon, A.J.; Mehonic, A.; Spiga, S.; Rana, V.; et al. Standards for the Characterization of Endurance in Resistive Switching Devices. *ACS Nano* **2021**, *15*, 17214–17231.
179. Park, S.O.; Jeong, H.; Park, J.; Bae, J.; Choi, S. Experimental Demonstration of Highly Reliable Dynamic Memristor for Artificial Neuron and Neuromorphic Computing. *Nat. Commun.* **2022**, *13*, 2888.
180. Wu, Y.; Wang, Q.; Wang, Z.; Wang, X.; Ayyagari, B.; Krishnan, S.; Chudzik, M.; Lu, W.D. Bulk-Switching Memristor-Based Compute-In-Memory Module for Deep Neural Network Training. *Adv. Mater.* **2023**, *35*, 2305465.
181. Liu, C.; Tiw, P.J.; Zhang, T.; Wang, Y.; Cai, L.; Yuan, R.; Pan, Z.; Yue, W.; Tao, Y.; Yang, Y. VO₂ Memristor-Based Frequency Converter with In-Situ Synthesize and Mix for Wireless Internet-of-Things. *Nat. Commun.* **2024**, *15*, 1523.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.