

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}, Junbin Long ^{1,2} and Zhong Sun ^{1,2,3,*} 

- ¹ Institute for Artificial Intelligence, Peking University, Beijing 100871, China; zhengmiao@stu.pku.edu.cn (Z.M.); saitao.zhang@stu.pku.edu.cn (S.Z.); fdhcc@stu.pku.edu.cn (C.H.); yongxiang.li@stu.pku.edu.cn (Y.L.); yubiao_luo@pku.edu.cn (Y.L.); shiqing.wang@stu.pku.edu.cn (S.W.); jblong@stu.pku.edu.cn (J.L.)
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, as programmable resistive switching devices, offer two fundamental computational modalities for artificial intelligence: static conductance for parallel data processing and dynamic switching for temporal, logical, and stochastic operations. This Review systematically distinguishes these two modalities and evaluates their respective hardware implementations. In terms of our review scope, we first examine how static conductance modality is exploited in analog matrix computing, which encompasses matrix–vector multiplication and matrix equation solving, and discuss how these primitives enable efficient neural network inference and training. Second, we survey dynamic switching modality and its 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 circuit 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



Academic Editor: Nikolay Hinov

Received: 19 March 2026

Revised: 29 April 2026

Accepted: 7 May 2026

Published: 10 May 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

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 (ML), neural networks (NNs), and 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 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 reduce 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]. While several recent reviews have examined memristor-based computing, most address static conductance or dynamic switching in isolation, or discuss both without an explicit structural division (Table 1). Notably, Mannocci et al. (2023) introduced a pioneering static/dynamic framework [9]. This review systematically organizes memristor-based AI hardware around these two intrinsic physical modalities. Beyond consolidating recent breakthroughs, we substantially expand the coverage to include previously underrepresented paradigms, such as closed-loop analog matrix computing (AMC) for second-order NN training, practical implementations of stateful logic and stochastic computing. Integrated with updated co-design strategies for modern architectures like Transformers, this article provides a timely, application-centric roadmap that explicitly leverages the dual physical modalities of memristors for next-generation AI systems. As illustrated in Figure 1, their functionality is fundamentally characterized by two key modalities: (i) static conductance—the ability to set and retain a multi-level conductance that can represent matrix elements; and (ii) dynamic switching—the nonvolatile or volatile time-dependent resistive transitions intrinsic to the devices.

Table 1. Comparison of recent reviews on memristor-based neuromorphic computing.

Attribute	Prior Paper	Application
Static Conductance	Memory devices and applications for in-memory computing [5]	NN, scientific computing, combinatorial optimization
	In-Memory Vector-Matrix Multiplication in Monolithic Complementary Metal–Oxide–Semiconductor–Memristor Integrated Circuits: Design Choices, Challenges, and Perspectives. [10]	NN, machine learning
Dynamic Switching	Emerging Materials and Computing Paradigms for Temporal Signal Analysis [11]	Reservoir computing, spatiotemporal signal detection, STDP
Covering both but unstructured	Neuromorphic computing using non-volatile memory [12]	NN, STDP, stateful logic
	In-memory computing with resistive switching devices [3]	Stateful Logic, NN, stochastic computation
Both as structured	In-memory computing with emerging memory devices: Status and outlook [9]	NN, Closed-loop AMC circuits, STDP, reservoir computing, spatiotemporal signal detection
	This Review	NN, Closed-loop AMC circuits, stateful logic, attractor networks, reservoir computing, spatiotemporal signal detection, STDP, stochastic computation

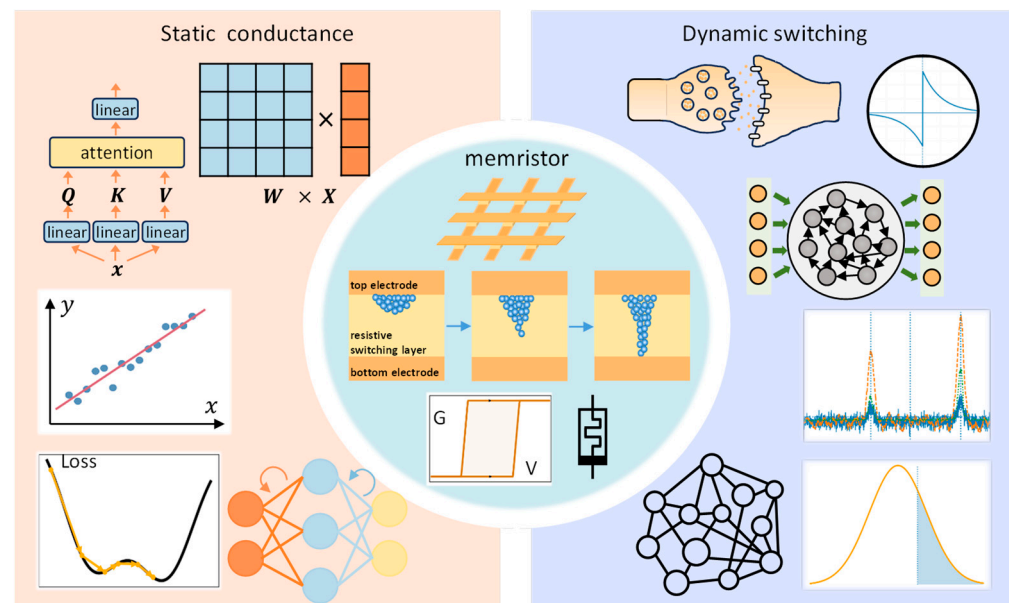


Figure 1. Memristor fundamentals and AI applications: physical mechanism, computing primitives enabled by static conductance and dynamic switching.

When exploiting the static conductance of memristors arranged in crossbar arrays, a primary application is AMC, which enables massively parallel, low-energy operations. The first major pathway of AMC is matrix-vector multiplication (MVM), the fundamental operation in fully connected, convolutional, and attention-based layers of NNs [13]. The second pathway extends AMC to the direct, one-step solution of matrix equations [14]. 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. To systematically exploit these dynamics for AI, their applications can be methodically classified into four primary physical paradigms [12,15–17]: (1) deterministic nonvolatile switching: leveraging abrupt, nonvolatile state transitions to implement stateful logic and associative attractor networks; (2) volatile relaxation switching: utilizing the spontaneous decay and fading memory of volatile memristors to construct reservoir computing substrates and spatiotemporal sensors; (3) cumulative nonvolatile switching: exploiting the pulse-driven, incremental conductance modulation in nonvolatile memristors to emulate biological spike-timing-dependent plasticity (STDP); and (4) stochastic switching: harnessing intrinsic switching randomness and thermal noise as physical entropy sources for stochastic computing.

This Review frames memristor-based AI hardware around these two complementary modalities—static conductance and dynamic switching. After an overview of device physics and performance metrics (Section 2), we examine static conductance approaches for analog matrix computations (Section 3) and dynamic-switching approaches for temporal, logical, and stochastic paradigms (Section 4). Finally, we synthesize open challenges and highlight algorithm-hardware co-design strategies to guide future efforts in developing robust, energy-efficient memristor-based AI systems (Section 5).

2. Memristor Background

The memristor concept was first proposed by Leon Chua (1971) and later experimentally demonstrated in nanoscale two-terminal devices [18,19]. In practice, most hardware implementations adopt a metal–insulator–metal (MIM) structure, where a thin active layer

is positioned between the two electrodes. The active material and electrode interfaces determine the dominant switching behavior and thus the device's electrical characteristics.

The switching behavior of modern memristors is commonly grouped into two broad categories. Nonvolatile resistive switching includes valence-change mechanism (VCM), electrochemical metallization (ECM), and phase-change memory (PCM). In VCM and ECM devices, resistance modulation is governed by field-driven ion migration and filament formation or rupture [20–25], whereas PCM relies on reversible amorphous–crystalline phase transitions in chalcogenide materials [26–28]. Volatile memristors, by contrast, relax back to a high-resistance state after the stimulus is removed and are therefore better suited to temporal processing and transient dynamics [29–31]. Representative examples include diffusion-mediated memristors, Ovonic threshold switching elements, and VO₂-based Mott devices. Importantly, the same memristor-based platform can often be tuned toward persistent weight storage or transient switching behavior through material engineering, interface design, current compliance, and pulse protocols [6]. Notably, the distinction between static conductance and dynamic switching is application-driven rather than material-driven. 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. Nonvolatile memristors (e.g., VCM, ECM, and PCM) typically exhibit on/off ratios ranging from 10² to 10⁵ and endurance spans from 10⁴ to over 10¹⁰ cycles [20–26]. Their switching energies generally fall within the femtojoule (fJ) to picojoule (pJ) regime, with PCM typically operating at the higher pJ end due to its thermally driven phase transition [5,26]. Crucially, due to the stochastic nature of filament formation and phase changes, these devices often suffer from noticeable cycle-to-cycle and device-to-device variability, posing a primary challenge for high-precision weight mapping [6]. In contrast, volatile memristors are characterized by spontaneous relaxation times (nanoseconds to milliseconds) and extremely high endurance (10¹⁰ to >10¹² cycles) as they avoid permanent structural rupture [17,31]. They feature ultra-fast switching (<30 ns) and inherently provide stochastic switching probabilities that are highly beneficial for temporal and probabilistic AI workloads [32]. 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 [33–38]. To aid readability, the principal symbols used in the mathematical derivations are listed in Appendix A (Table A1), which also clarifies notations that may vary across sections. As shown in Figure 2a, the array stores programmable conductances A_{ij} at each crosspoint, representing the weight matrix A . When a voltage vector x is applied to the word lines (WLs), the current flowing through each device is $I_{ij} = G_{ij} \cdot x_j$. 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: $I_i = \sum_j G_{ij} x_j$. 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 circuits, including transimpedance amplifiers (TIAs) and analog-to-digital converters (ADCs).

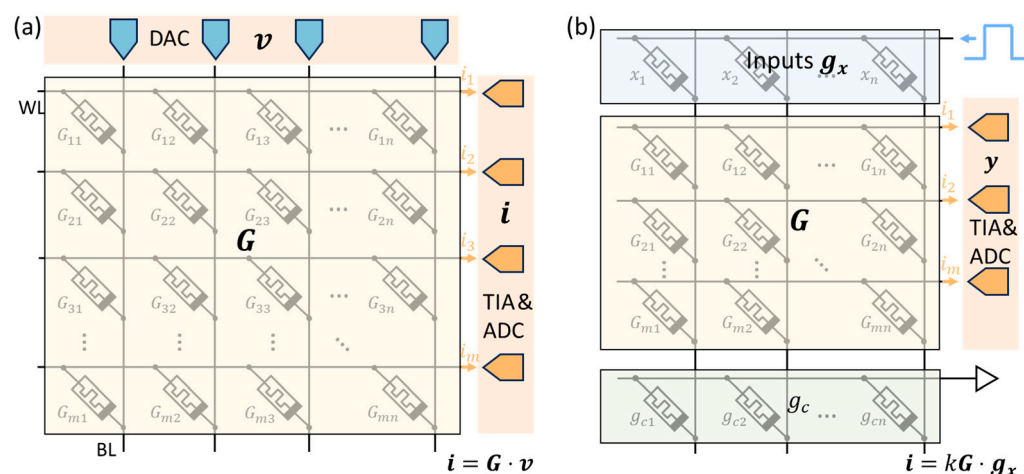


Figure 2. MVM with memristor crossbar arrays. (a) Crossbar MVM principle. Each weight element is encoded as a memristor conductance G_{ij} . Applying the input voltage vector on the BLs produces WL currents $i_i = \sum_j G_{ij}x_j$, which implement the MVM in one physical step. (b) Dual in-memory computation scheme. Inputs are encoded as conductances in another row of the array, so that both weights and inputs are processed in-memory and the multiplication is performed with reduced peripheral data movement.

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 IMC method where the input vector is also represented as conductance states within the array, as shown in Figure 2b [39]. In this approach, a voltage pulse is applied to the word-line corresponding to the row that stores the input vector $\mathbf{g}_x$. 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: $V_{BLj} = kg_x$, 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 NN architectures. However, its practical implementation is constrained by limited computational accuracy due to device non-idealities, as well as by significant overhead introduced by peripheral circuits, including DACs and ADCs. These combined bottlenecks restrict the effective numerical resolution of practical implementations to approximately 5–8 bits for target workloads [40].

3.2. Applications of MVM in Neural Networks

Memristor crossbars offer a versatile MVM primitive capable of accelerating a wide range of NN 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 G_{ij} , and the input vector $\mathbf{x}$ is applied as WL voltages; the resulting BL currents represent the output activations [41–45]. Because dense MVMs dominate many AI workloads, crossbars deliver high throughput by per-

forming 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 [41]; Dot-Product Engine-type architectures explore weight encoding, negative-value mapping, and precision decomposition [46]; and mixed-precision approaches perform most dot products in low-precision analog and correct residuals digitally, achieving efficient energy–accuracy tradeoffs [34]. FCNs represent a well-established and promising target for crossbar-based acceleration, with substantial progress already demonstrated in practice.

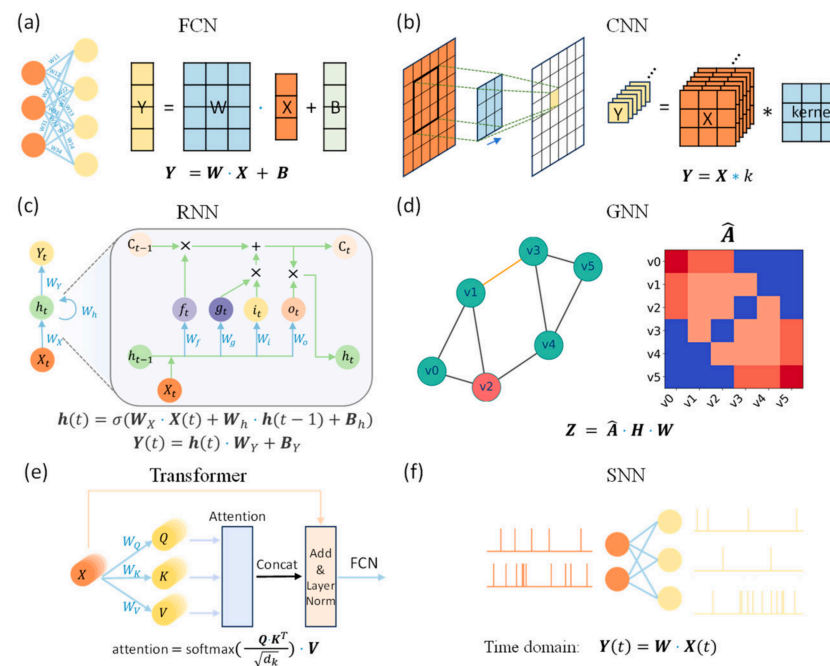


Figure 3. MVM mappings for common AI operators. (a) FCN: each output neuron performs a weighted sum over all inputs via crossbar MVM. (b) CNN: convolution is decomposed into repeated MVMs over flattened input patches and kernels. (c) RNN and LSTM: time-step matrix operations implemented by sequential 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. In the diagrams and equations, the symbol ‘ $\cdot$ ’ represents matrix multiplication, ‘ $*$ ’ denotes the convolution operation, and arrows indicate the data flow and sequence of computation.

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 [47]. Consequently, convolutions map naturally to memristor crossbars by storing flattened kernels as column-wise conductance vectors and applying input patches as row voltages [40,48–51]. 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 [9,46,52]. 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 $W \cdot x_t$ and $W \cdot h_{t-1}$ for the gates and candidate updates in an LSTM cell, as illustrated in Figure 3c [53]. Mem-

ristor 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 [54–58]. 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 [54,56]. 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 $H(l+1) = \sigma(\hat{A} \cdot H(l) \cdot W(l))$, where $\hat{A}$ is an adjacency matrix and $H(l)$ is the node-feature matrix of the l -th layer, as illustrated in Figure 3d [59]. Because adjacency matrices are sparse and access patterns irregular, directly mapping GNN operations onto dense crossbars results in low utilization and significant energy inefficiency [60,61]. 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 [62,63]. These works demonstrate that with sparsity-aware scheduling and heterogeneous compute support, crossbar-based GNN accelerators can achieve competitive performance [64–66].

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 [67]. Unlike many CNN kernels, the operands in attention layers vary dynamically at runtime. For modern large language models (LLMs), particularly in long-context generative tasks, the primary bottleneck has shifted from initial projections to the management and reuse of the key-value (KV) cache. This shift introduces severe memory-wall pressure, including frequent writes of intermediate matrices and substantial ADC/DAC overhead. To address these issues, recent memristor-based CIM architectures emphasize hardware–algorithm co-design. Sequence blocking and tiling strategies partition long contexts to align with crossbar dimensions, while matrix decomposition and pipelined dataflows optimize attention computation efficiency [68,69]. In addition, KV cache compression methods reduce the memory footprint of long-context inference [70]. Several memristor-based attention accelerators such as Attar execute attention in-memory, reducing write-backs and memory-transfer overhead [71–75]. Collectively, these co-optimized strategies demonstrate that memristor crossbars can efficiently accelerate Transformer.

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 memristors inherently exhibit thresholding and short-term dynamic behaviors, enabling compact hardware neuron implementations that integrate seamlessly with synaptic crossbars [76]. 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 [77]. 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. Closed-Loop Matrix Equation Solving in AMC

Memristor crossbars can also be employed to implement efficient closed-loop matrix equation solvers [11,78]. 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 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, generalized inverse (GINV), eigenvector extraction, and sparse approximation. Throughout this section, we consistently map the mathematical variables to their circuit counterparts: the coefficient matrix A is encoded as the conductance matrix G , the known vector y is applied as the input current i , and the solution vector x is read out as the output voltage v .

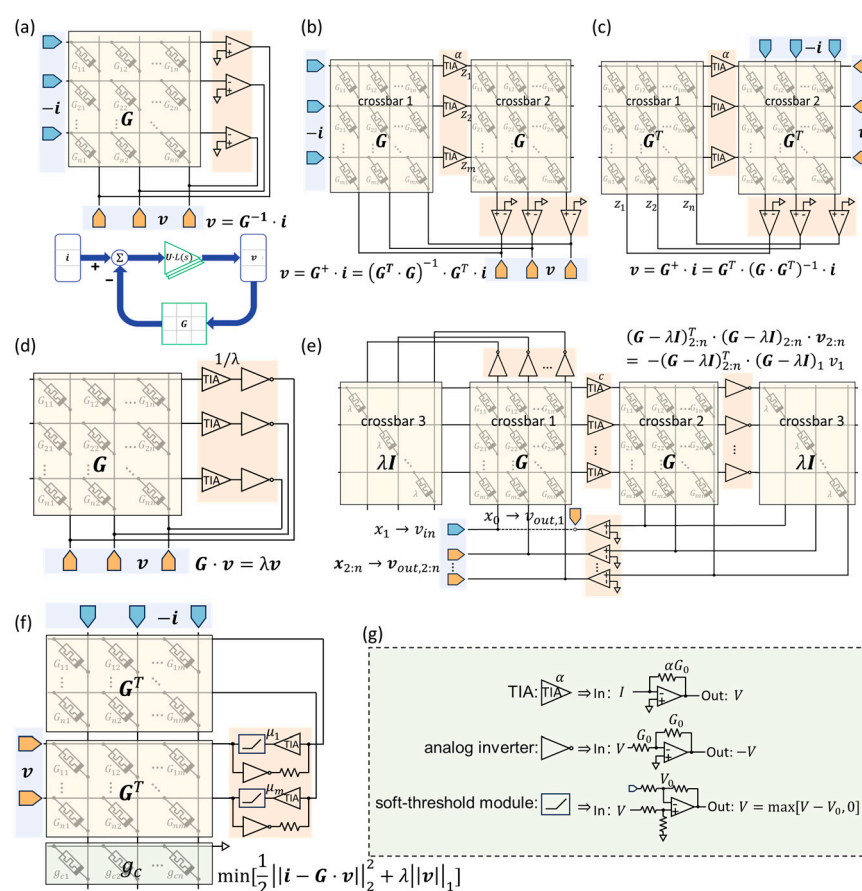


Figure 4. Closed-loop AMC circuits. (a) Matrix inversion circuit: The upper panel shows the crossbar is embedded in a global feedback loop with op-amps to solve $G \cdot v = i$ in the analog domain; the lower panel presents the block diagram of this closed-loop matrix inversion circuit, Reproduced with permission of Ref. [79], Copyright 2020 IEEE. (b) Left GINV circuit: a two-array feedback configuration yields $v = (G^T \cdot G)^{-1} \cdot G^T \cdot i$. (c) Right GINV circuit: topology using transposed matrix and feedback to compute $v = G^T \cdot (G \cdot G^T)^{-1} \cdot i$. (d) Eigenvector circuit: feedback configuration that enforces $G \cdot v = \lambda v$ to obtain eigenvectors for a chosen eigenvalue λ . (e) Modified-GINV circuit: a four-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.

Matrix Inversion (Figure 4a). The inversion circuit (INV) solves the linear system $A \cdot x = y$ for an $n \times n$ matrix A . The crossbar that encodes G to conductance values and 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 is applied to the op-amp inputs, the closed-loop feedback enforces the corresponding steady-state solution

$$x = A^{-1} \cdot y. \quad (1)$$

These closed-loop feedback dynamics are depicted in the block diagram of Figure 4a (bottom). The convergence time of this topology depends on the spectral properties of A (in particular its smallest eigenvalue), rather than directly on matrix size [79]. 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 (Figure 4b,c). Non-square or rank-deficient problems require a 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 A and two sets of amplifiers form the feedback network. When the input current vector y is applied, the steady-state solution is

$$x = (A^T \cdot A)^{-1} \cdot A^T \cdot y = A^+ \cdot y. \quad (2)$$

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

$$x = A^T \cdot (A \cdot A^T)^{-1} \cdot y = A^+ \cdot y. \quad (3)$$

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

Eigenvector Circuit (Figure 4d). The eigenvector circuit directly enforces

$$A \cdot x = \lambda x \quad (4)$$

via a positive feedback configuration, with the TIA conductance encoding the eigenvalue λ of matrix A [80]. The steady-state outputs are the eigenvectors associated with λ . This analog approach enables direct extraction of principal directions, which can accelerate principal component analysis (PCA) and spectral methods used in AI pipelines [81].

Modified-GINV Circuit (Figure 4e). Figure 4e illustrates an AMC circuit designed to solve for all eigenpairs (eigenvalues and eigenvectors) of a matrix [82]. The topology maps the target matrix A and the diagonal matrix λI onto two memristor arrays, leading to the matrix relation:

$$(A - \lambda I)_{2:n}^T (A - \lambda I)_{2:n} x_{2:n} = -(A - \lambda I)_{2:n}^T (A - \lambda I)_1 x_1. \quad (5)$$

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 and monitoring the output voltage of the op-amp in the disconnected branch for a characteristic peak, the circuit identifies the eigenvalue of G . This approach enables the extraction of the

complete set of eigenpairs for a given matrix, which can accelerate PCA, spectral clustering and graph learning.

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

$$\min_x L(x) = \min_x \left[1/2 \|y - A \cdot x\|_2^2 + \beta \|x\|_1 \right] \quad (6)$$

where β controls the sparsity penalty [83]. In this design, the compensation conductance g_c ensures conductance sum of each column equal to a constant, thus the crossbar output current proportional to $G^T(G \cdot v - i)$. That current is converted by TIAs to node voltages μ , and the sparsity output x is obtained through soft-thresholding nonlinearity (Figure 4g). The analog inverters are applied for cancelling the diagonal contribution of the Gram product $G^T \cdot G$. The circuit performs the locally competitive algorithm (LCA):

$$\frac{d\mu}{dt} = \frac{1}{\tau} \left[-\mu + A^T \cdot y - (A^T \cdot A - I) \cdot x \right]. \quad (7)$$

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

By constructing a closed-loop architecture between the memristor crossbar array and op-amps, AMC circuits can achieve rapid solutions to complex linear-algebra problems fast, typically converging within nanoseconds to microseconds [79–81]. Thereby, they provide efficient hardware primitives for AI tasks such as regression, optimization, and NN training.

3.4. Applications of Closed-Loop AMC Circuits

3.4.1. Linear/Logistic Regression

As a fundamental ML model, linear regression is widely used for regression and predictive analysis across diverse fields, including biology, statistics and finance [84]. 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 [85]. 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 [85]. 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 FCN. It is also commonly employed as the final output layer in multilayer NNs, 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 NN for modified national institute of standards and technology (MNIST) digit recognition—and can effectively accelerate reservoir computing training [85–87]. Previous studies have also demonstrated that such circuits exhibit significant performance advantages across multiple benchmark metrics [88–92]. Their computational speed can further be enhanced through parameter co-optimization [93].

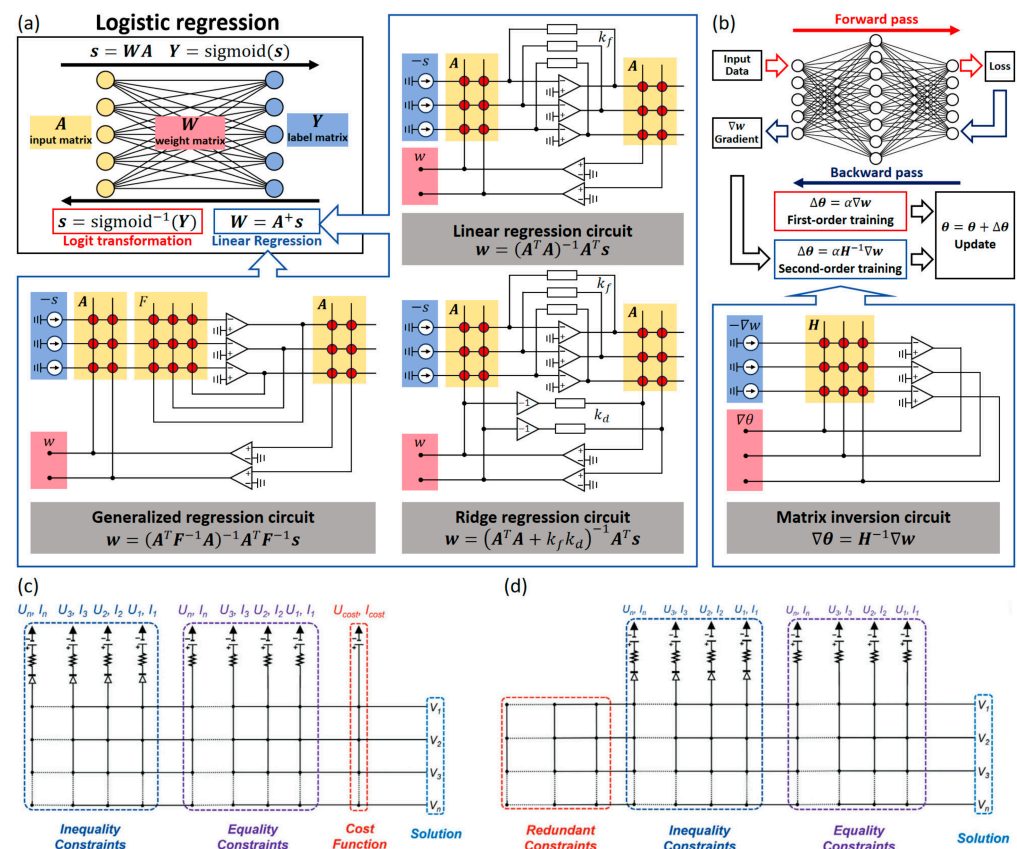


Figure 5. Applications of AMC Circuit. (a) Based on the logit transformation, logistic regression can be transformed into linear regression and accelerated through linear regression circuit. Generalized regression and ridge regression circuits can further expand the scope of accelerated regression problems. (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. In (c,d), the arrows represent the optimization circuits, while the dashed boxes delineate the functional mapping of different constraints onto the array. Reproduced with permission of Ref. [94], 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 [95]. Meanwhile, a ridge regression circuit has been developed to implement the regularization term by adding an additional negative feedback branch [96]. Although applications of generalized and ridge regression circuits in NN 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 NN training [97].

3.4.2. Second-Order Neural Network Training

In NN 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 [98]. However, on conventional computing platforms such as graphics processing unit (GPU) and central processing

unit (CPU), 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 second-order training accelerators-based on matrix inversion circuits achieve significant speed and energy efficiency improvements over conventional GPU and PipeLayer baselines [99,100] (see Discussion for quantitative comparisons). In parallel, inspired by the intrinsic mechanisms of AMC, the thermodynamic approach to computation has emerged as a promising paradigm for solving mathematical problems [101]. Its core principle lies in exploiting the stochastic dynamic behavior of physical systems governed by the interplay of conservative, dissipative, and fluctuating forces. Thermodynamic computing has been shown to accelerate tasks such as matrix determinant evaluation and solving matrix equations [102,103]. Furthermore, it has been applied to Bayesian inference and to accelerating various second-order NN training algorithms, including natural gradient descent and Kronecker-factored approximate curvature (KFAC) [104–106]. Notably, these thermodynamic approaches have also demonstrated superior speed and energy efficiency compared with first-order methods such as stochastic gradient descent (SGD) and Adam.

3.4.3. Linear Programming and Quadratic Programming

The parallelism of closed-loop AMC circuits 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 ML, robot kinematics and drone control [107]. LP and QP problems can be formulated as minimizing an objective function $x^T Ax + c^T x$ subject to equality and inequality constraints, where a QP problem reduces to an LP problem when the Hessian matrix A 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 [108,109]. 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 time that is typically small. Furthermore, optimized circuit structures can extend AMC solvers to tackle more complex problems, including mixed-integer programming [110,111].

By fully exploiting the inherent differential characteristics of feedback loops, several studies have proposed LP solvers-based on RNN architectures and closed-loop AMC solvers [112–115]. As shown in Figure 5c,d, the ultra-fast hybrid computing system integrates tuning, control, optimization, readout circuits, and memristor arrays [94]. By directly mapping the problem constraints and objective function to the memristor array conductances and input voltage values, the solution can be obtained rapidly at the output port, reducing the computational complexity of LP and QP optimization. In application scenarios such as miniature aerial vehicle control and edge computing for clinical image analysis, the closed-loop AMC solver demonstrates ultra-fast response speeds and massive energy efficiency improvements compared to standard microcontrollers units (MCU), substantially alleviating computational pressure in these domains.

3.4.4. PCA and Spectral Clustering

PCA is a classical linear dimensionality reduction technique in ML. 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. The closed-loop AMC circuit in Figure 4e provides a

hardware acceleration solution by mapping the covariance matrix C onto array conductances and sweeping the λ value in the diagonal array [82,116]. When λ coincides with an eigenvalue, the op-amp output exhibits a characteristic peak, while the voltage vector 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 $L = D - A$ [117]. 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 with ultra-low latency, directly outputting all required eigenvectors. Because the circuit solves the complete set of eigenpairs, it is particularly suited for spectral clustering, which requires multiple non-dominant eigenvectors. The parallelism and low-latency characteristics of AMC, combined with memristor IMC, 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 graph learning applications [118–120].

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). To systematically harness these behaviors for AI, we first establish a unifying framework that maps intrinsic switching dynamics to their corresponding computational paradigms (Table 2). Specifically, deterministic nonvolatile switching characterizes abrupt state transitions harnessed for stateful logic (Section 4.1) and attractor network (Section 4.2). Conversely, volatile relaxation switching utilizes the spontaneous decay to provide a natural fading memory essential for reservoir computing (Section 4.3). Regarding synaptic learning, cumulative nonvolatile switching relies on cumulative, history-dependent conductance modulation to implement local plasticity rules like STDP (Section 4.4). Finally, stochastic switching exploits the probabilistic nature of ion migration or phase transition to offer a physical substrate for stochastic computing (Section 4.5). This unifying framework explicitly contrasts the required physical dynamics, corresponding device types, and system-level applications.

Table 2. Taxonomy of Dynamic Switching Applications in AI.

Required Physical Dynamics	Application Paradigm	Typical Device Type	Core AI Function
Deterministic nonvolatile switching (abrupt SET/RESET)	Stateful logic & attractor networks	Nonvolatile	In-memory Boolean logic, associative memory
Volatile relaxation switching (fading memory)	Reservoir computing & spatiotemporal signal detection	Volatile	Temporal feature extraction, short-term memory

Table 2. Cont.

Required Physical Dynamics	Application Paradigm	Typical Device Type	Core AI Function
Cumulative nonvolatile switching (incremental modulation)	STDP	Nonvolatile	Unsupervised local learning, synaptic weight update
Stochastic switching (intrinsic noise)	Stochastic computing	Volatile/nonvolatile	Bayesian inference, random number generation

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 [121–126]. 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, reducing 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 [121]. 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:

$$V_P = \frac{V_{\text{con}}g_A + V_{\text{set}}g_Y}{g_X + g_Y + g_L}. \quad (8)$$

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 voltage across Y 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 [122]. 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

$$V_P = \frac{V_A g_A + V_B g_B}{g_A + g_B + g_Y}. \quad (9)$$

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 [123]. As shown in Figure 6c, memristors A and B encode the logic inputs, and the output mem-

ristor Y stores the computation result. Applying KCL at the intermediate node, the voltage across the output memristor can be expressed as a weighted sum of input contributions:

$$V_Y - V_P = w_A x_A + w_B x_B + b. \quad (10)$$

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.

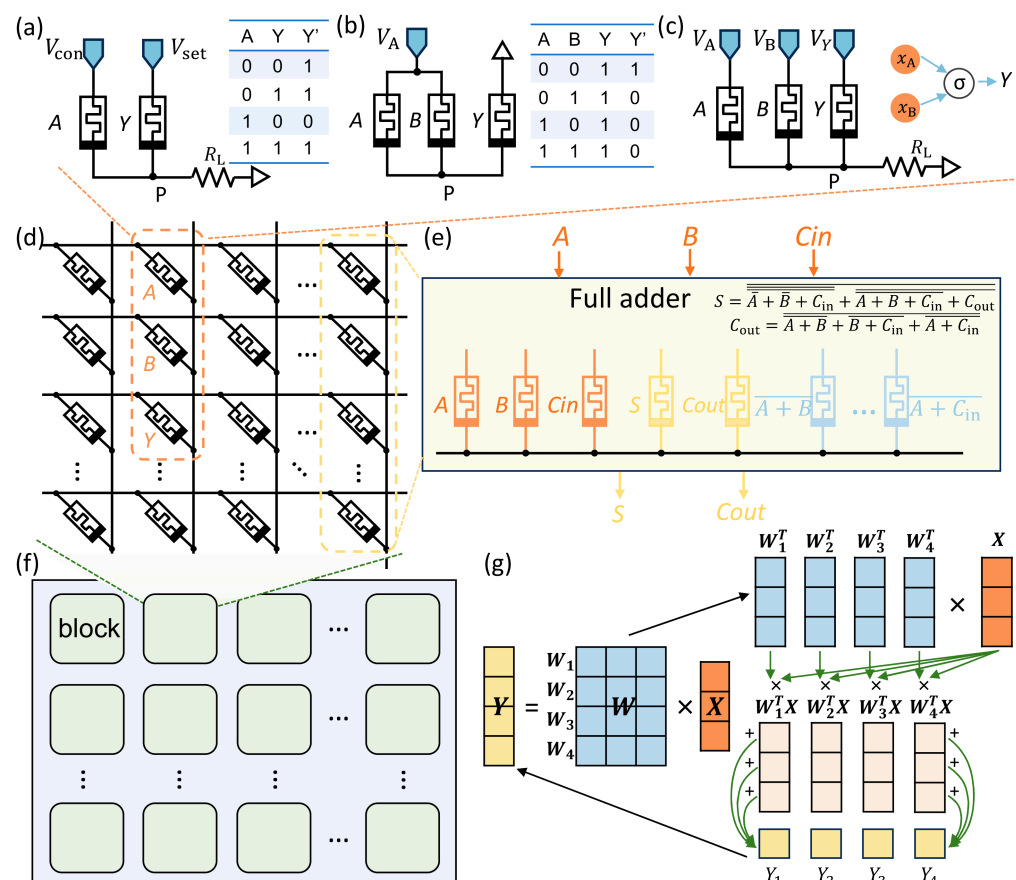


Figure 6. Stateful logic with memristors and its AI applications. (a) IMP primitive implemented with memristors. (b) MAGIC circuit for NOR. (c) Stateful neural network circuit acting as a single-layer perceptron, where the arrow denotes the signal path from inputs to output. (d) Mapping between crossbar 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 memristor 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 Figure 6d,e. Researchers have proposed MAGIC-based designs for fixed-point multiplication and for floating-point multiplication adapted to crossbar arrays [127]. Such adders and multipliers can be composed to implement MVM primitives, which are central to NN 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 IMC architectures are built upon these MAGIC-based stateful logic primitives to accelerate AI workloads. As shown in Figure 6f,g, FloatPIM is a representative architecture that decomposes high-precision operations into sequences of bitwise stateful gates and maps them efficiently onto crossbar arrays [128]. FloatPIM and related systems implement CNN and FCN by bit-serial or hybrid mappings, trading the number of bitwise cycles for reduced data movement and high array parallelism [128–131]. Other architectures—ConvPIM, RIME, ReHy, and DUAL—apply similar principles for CNNs, transformers, clustering, and other ML tasks [132–135]. 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 dot product with stateful digital logic for activations, normalization, and control, thus maximizing the strengths of both paradigms [136].

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 [137]. 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.

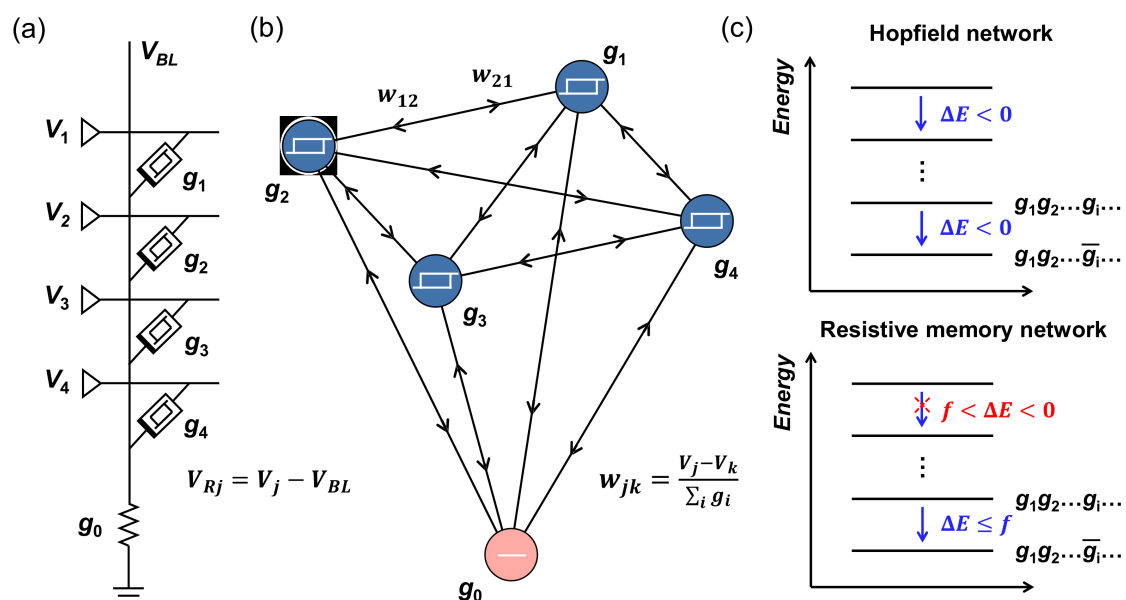


Figure 7. Attractor Network. (a) Resistive-memory column circuit with a memristor and load resistor. (b) Abstracted RNN model of the memristor circuit, showing the weight definition, where the arrows indicate the mutual interactions between different memristor nodes. (c) Illustration of conditions for a 1-bit flip in the memristor network and comparison to a Hopfield network, where the arrows represent the transitions of system energy states caused by a state flip. Reproduced with permission of Ref. [137], Copyright 2024 Nature Communications.

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 [12,138–140]. 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 networks comprise simple memristor arrays, where feedback operates continuously under KCL and accelerates attractor dynamics convergence.

4.3. Reservoir Computing and Spatiotemporal Signal Detection

Volatile memristors are particularly well suited for temporal and spatiotemporal signal processing. Their internal ionic or defect dynamics provide a controllable fading memory and intrinsic nonlinearity. The fading memory originates from ion migration and subsequent relaxation processes, which cause device responses to decay over characteristic time scales [86,141,142]. 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 volatile memristors natural candidates for hardware reservoir computing [143]. 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 dynamic 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 volatile memristors to build a compact reservoir. Input sequences were encoded as pulse trains, and the transient responses of the devices were used as reservoir states. A linear readout was trained to classify framed MNIST patterns and to predict nonlinear time series, with the same memristor 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 (Figure 8a,b) [144]. To further enhance state richness and resource efficiency, subsequent work combined simple masking with parallel device sampling [145]. This masked-parallel scheme generates virtual nodes from a limited number of physical devices through time-multiplexing. This approach achieved strong results on spoken-digit recognition and chaotic-series prediction (Figure 8c–h). However, this method faces a fundamental trade-off between hardware footprint and state richness. While generating virtual nodes conserves physical area, scalability remains inherently constrained. Increasing the mask length to expand state dimensionality dilutes the temporal feedback strength between sequential states [145]. This limits the maximum achievable complexity. Therefore, masked-parallel architectures suit resource-constrained edge scenarios [145]. True multi-device physical reservoirs remain essential for highly complex temporal tasks requiring massive parallelism and rich state topologies. The same research team later demonstrated a fully analog reservoir computing system in which volatile memristors serve as the reservoir and nonvolatile memristor arrays implement the readout layer. These

systems enabled low-power, real-time spatiotemporal processing such as ECG and gesture recognition [146].

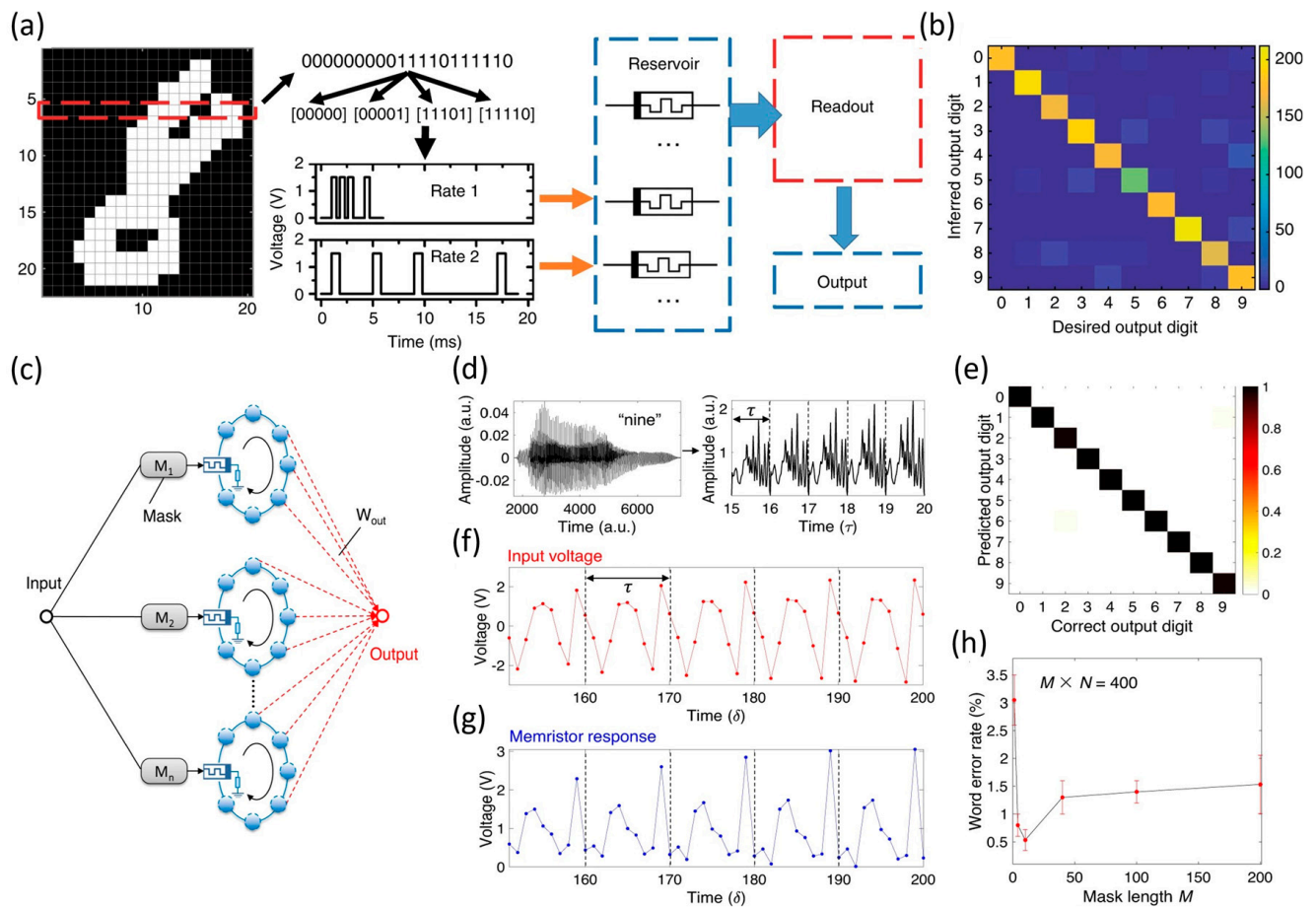


Figure 8. Reservoir computing (a,b) Schematic of a dynamic memristor reservoir system: input frames are encoded into pulse sequences and then applied to the reservoir; the resulting framed-digit classification results are shown. Reproduced with permission of Ref. [144], Copyright 2017 Nature Communications. (c–h) Masking and parallel single-device architecture for state expansion: (c) schematic of the dynamic memristor-based parallel reservoir-computing system; (d) spoken-digit recognition example, showing the input audio waveform; (e) time-multiplexing and masking procedure; (f) dynamic memristor responses recorded; (g) predicted results versus correct outputs for spoken-digit recognition; and (h) word-error-rate dependence on mask length. Reproduced with permission of Ref. [145], Copyright 2021 Nature Communications.

Beyond pure reservoir computing, volatile memristors have been integrated into adaptive neuromorphic perception pipelines [12,147–154]. 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 [147]. 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.

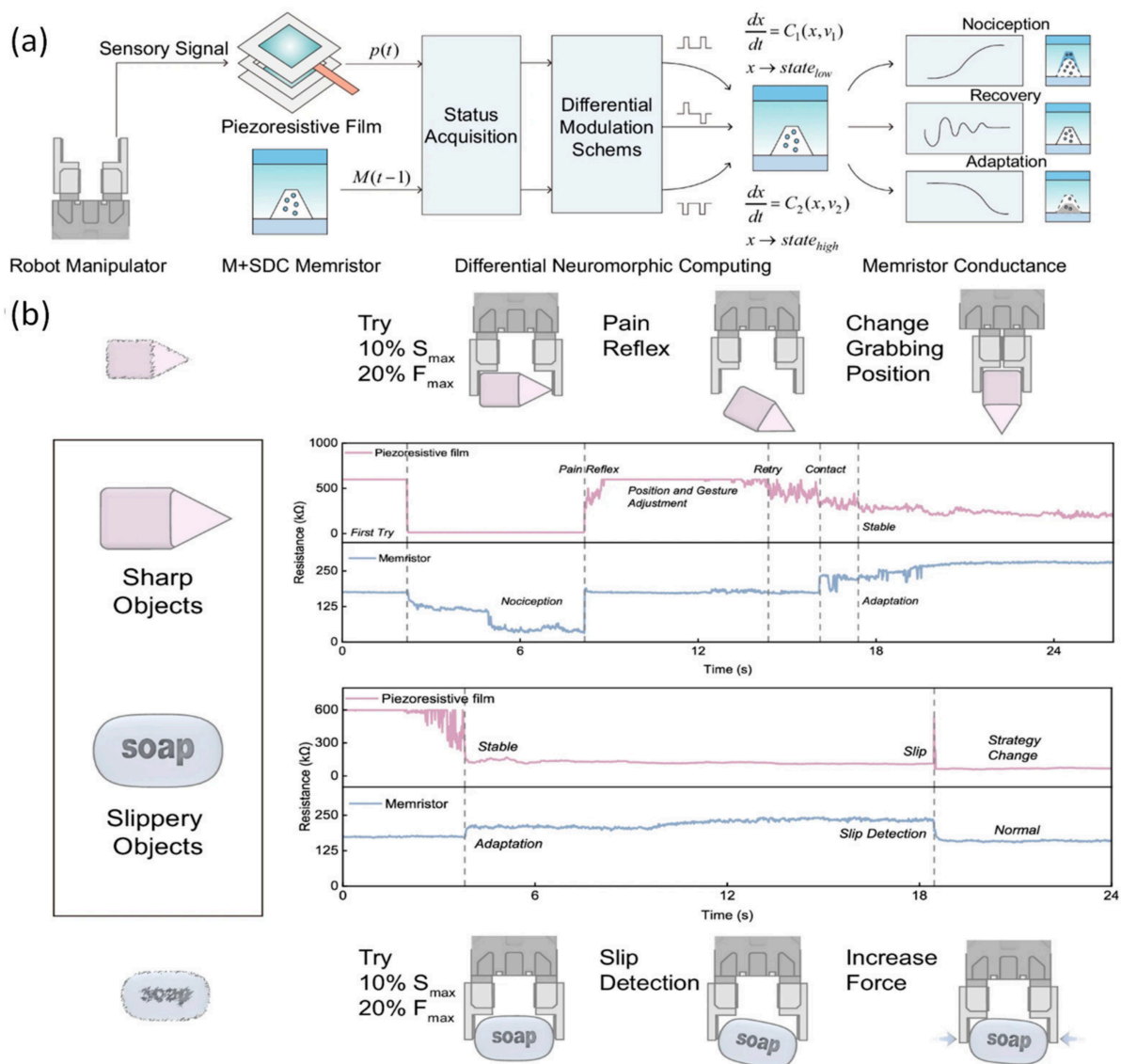


Figure 9. Spatiotemporal signal detection (a) Memristor-based near-sensor processing and adaptive response pipeline. The arrows indicate the transmission pathways of signals and pulses between components. (b) Representative task demonstrations showing grasping/scene extraction outcomes and the modulation of memristor conductance over the action duration. Reproduced with permission of Ref. [147], Copyright 2024 Nature Communications.

4.4. Spike-Timing-Dependent Plasticity

STDP is a local, timing-based synaptic learning rule [155,156]. 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

$$\begin{aligned} \Delta w &= A_+ \exp(-\Delta t / \tau_+) \text{ for } \Delta t > 0, \\ \Delta w &= -A_- \exp(\Delta t / \tau_-) \text{ for } \Delta t < 0. \end{aligned} \quad (11)$$

where $\Delta t = t_{pre} - t_{post}$. Positive Δt yields potentiation (LTP), and negative Δt yields depression (LTD) [155].

Hardware demonstrations have shown that memristors can implement STDP using simple spike protocols [157–166]. The first hardware evidence of memristor-based STDP was reported using a nonvolatile memristor as a synapse in a hybrid system that combined CMOS integrate-and-fire neurons with memristor-based synapses. In this system, the

neuron circuit converted relative spike timing into pulse-width information to induce STDP [158]. Building on this foundation, passive memristor crossbars were integrated with analog leaky-integrate-and-fire (LIF) silicon neurons to experimentally measure the canonical pair-STDP window (Figure 10a) [167]. This work provided 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 has been demonstrated with weight updates under three-spike protocols (Figure 10b) [168]. The observed behavior matches triplet model predictions and highlights the role of internal device traces in shaping plasticity.

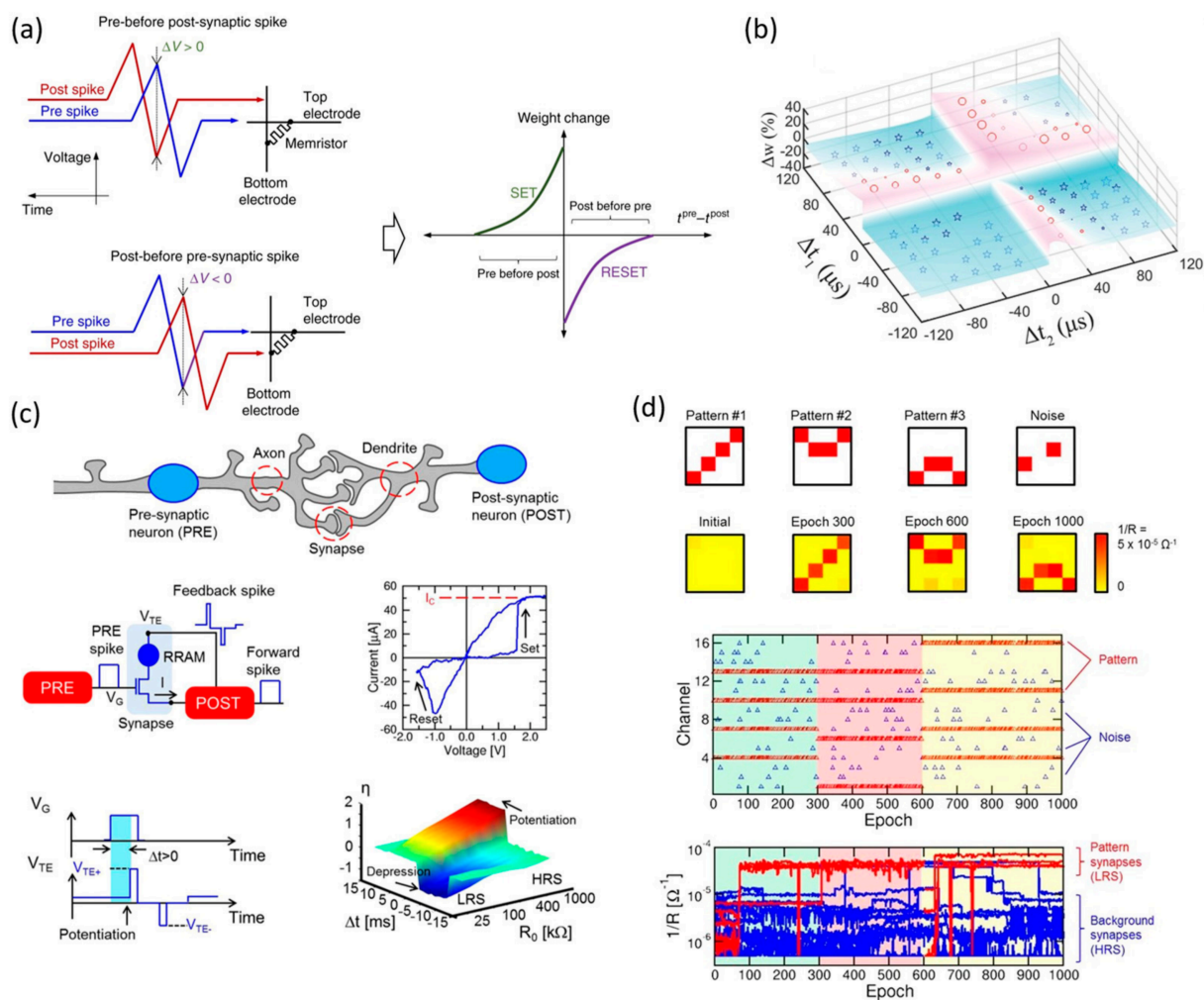


Figure 10. (a) Experimental STDP implementation and measured pair-STDP window: typical pre/post pulse shapes and resulting Δw versus Δt . Reproduced with permission of Ref. [167], Copyright 2018 Nature Communications. (b) Triplet-STDP: weight change under three-spike protocols and comparison with the triplet-STDP model. Reproduced with permission of Ref. [168], Copyright 2017 Advanced Functional Materials. (c) Hardware synapse and neuron module: circuit diagram of synapse interfaced with an LIF neuron. In the diagrams, the arrows indicate the state transitions of the memristor, and the red dashed line represents the compliance current limit. (d) Online pattern learning and tracking: evolution of synaptic weights and system performance on pattern-recognition/tracking tasks. Reproduced with permission of Ref. [169], Copyright 2018 Nature Scientific Reports.

At system scale, STDP has been used for online learning and pattern tracking. Memristor-based synapses were integrated with simple spiking neurons [169]. Continuous online experiments were conducted in which synaptic weights evolved via STDP while the network learned and tracked spatiotemporal patterns (Figure 10c,d). 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 on memristor-based 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 [157]. System and architecture studies emphasize multi-memristor synapses, selector/transistor integration, and pulse engineering as practical routes to scalability and reliability [161,164].

4.5. Memristor-Enabled Stochastic Computation

Memristors exhibit intrinsic, pulse-dependent stochasticity originating from ion migration, filament nucleation/rupture and phase-transition kinetics [170]. Two measurable statistics summarize this behavior. One is the discrete switching probability $p_{\text{switch}}(V, t)$, which quantifies the likelihood that a device transitions between states under a specific pulse [171–175]. The other is the post-pulse multilevel conductance distribution, characterizing the continuous spectrum of conductance states that a device may assume after programming [176–182].

This device-level randomness can be repurposed as a computational resource through two complementary approaches. The first primitive is probabilistic bits (p-bits). Under near-threshold excitation, a memristor produces a binary outcome whose mean output follows a sigmoidal dependence on the applied bias; thus, modulating the bias directly tunes 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 via the central limit theorem. 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. $\text{Cu}_{0.1}\text{Te}_{0.9}/\text{HfO}_2/\text{Pt}$ volatile memristors were demonstrated as reliable p-bit sources, with dynamic switching statistics systematically reported and a direct mapping established between pulse amplitude and sampling probability [171]. Small p-bit networks based on these devices were used to realize stochastic logic (Figure 11a,b). An in-memory deep Bayesian active-learning system was implemented [175]. This system leverages post-pulse conductance histograms, exploits device noise as the Langevin perturbation in mSGLD, and demonstrates improvements in labeling efficiency and energy consumption for active learning tasks (Figure 11c–e).

Additional works extend and validate these concepts across device types and applications. Versatile stochastic dot-product circuits have been demonstrated, exploiting controlled device and circuit variability to implement approximate probabilistic MVM operations suitable for edge inference [176]. Volatile 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 stochastic computing [173].

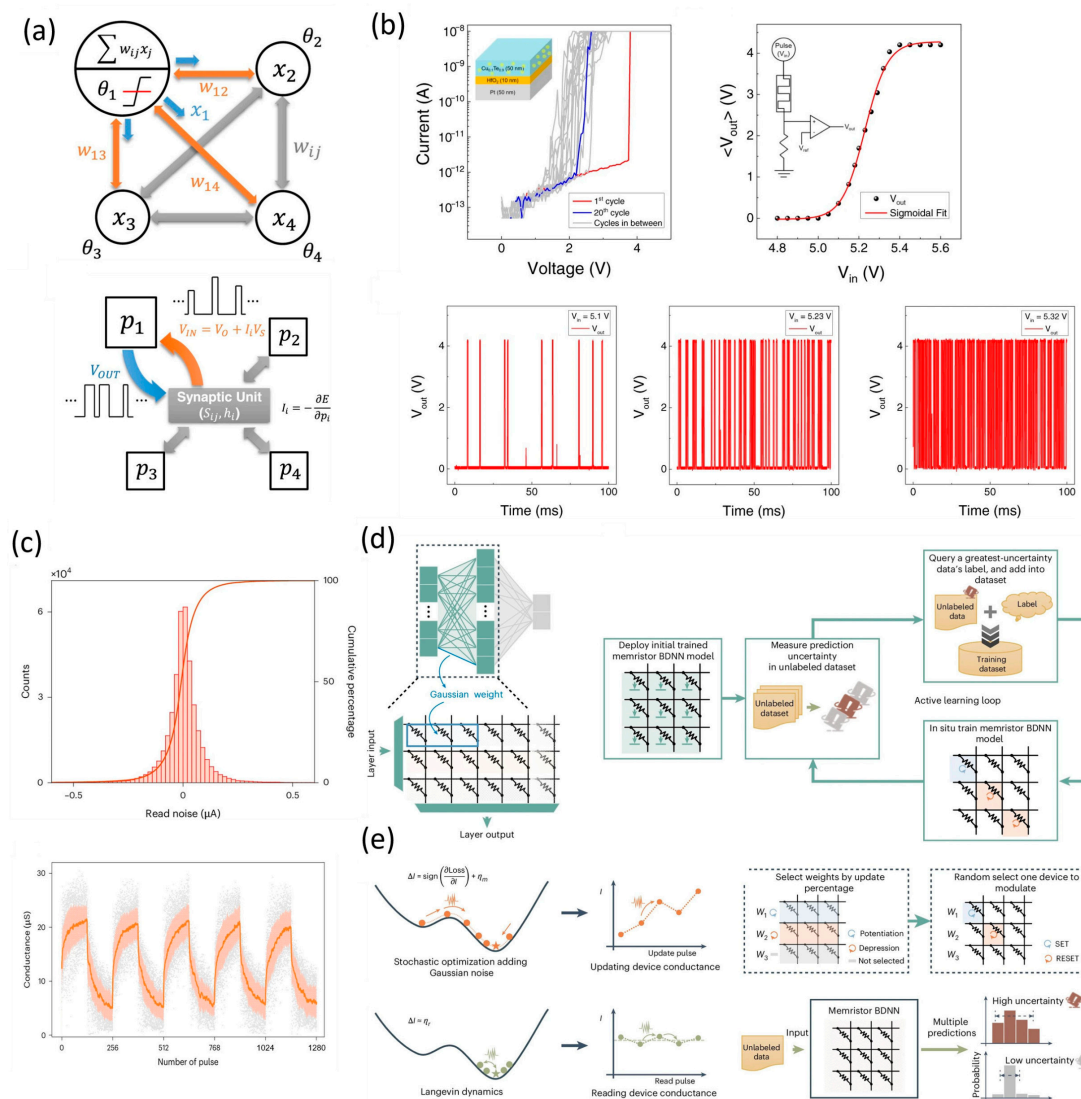


Figure 11. (a,b) Device threshold-switching statistics and p-bit logic mapping: (a) network architectures of the stochastic Hopfield and probabilistic p-bit systems (arrows indicate synaptic connections signal flow direction); and (b) a p-bit demonstration, including its I–V characteristics, sigmoidal transfer behavior, and output states under different input voltages. Reproduced with permission of Ref. [171], Copyright 2022 Nature Communications. (c–e) Conductance-change histograms, the mSGLD in-memory update loop, and system-level active-learning gains; (c) read-noise and conductance-fluctuation statistics of memristors; (d) in-memory Bayesian active learning and mSGLD workflow; and (e) weight-update and uncertainty-estimation results showing application-level benefits. Reproduced with permission of Ref. [175], Copyright 2024 Nature Computing Science.

5. Discussion

5.1. Functional Roles of Memristors Across AI Applications

Memristor-based accelerators for AI, whether leveraging static conductance for dense linear algebra or dynamic switching for temporal, logical, and stochastic operations, share a common hardware foundation and a common set of constraints. At the functional level, the applications summarized in Table 3 can first be distinguished according to whether memristors play synaptic or neuronal roles. Synaptic functions primarily store or modulate weights and therefore implement linear accumulation or plasticity mechanisms, including MVM, closed-loop AMC, reservoir computing, STDP, and stochastic computing based on continuous noisy weights. By contrast, neuronal functions implement nonlinear activation, thresholding, or state evolution, and thus include attractor networks, stateful logic, and

stochastic computing based on p-bit behavior. This distinction follows the general structure of NNs, where synapses provide weighted signal integration while neurons introduce nonlinear transformation and decision mechanisms.

Table 3. Application statistics of static conductance and dynamic switching.

Application	Functional Role	Typical Device	Key Feature	Conductance States & Precision	Latency/Time Constants	Randomness Role	Endurance Stress	Peripheral Criticality
MVM	Synapse	Nonvolatile memristor: VCM, PCM, nonvolatile ECM (e.g., Ag/Ge ₃₀ Se ₇₀ /W)	Highly parallel dot-products	Multilevel; medium precision (5–8 bits)	~10 ns	Harmful	Low (inference), high (training)	High (DAC/ADC, driver)
Closed-loop AMC circuits			Closed-loop for matrix computing (e.g., inversion)		50 ns to 10 μs (AMC topology-dependent)		High (second-order training), low (other tasks)	High (DAC/ADC, driver, op-amp)
Stateful logic	Neuron		Deterministic SET/RESET, binary	~10 ns	Beneficial	High (frequency switching)	Medium (pulse driver, selector)	
Attractor networks				~100 ns				
Stochastic computing—continuous noisy weights			Analog statistical conductance	Multilevel; lower precision				~10 ns
STDP	Synapse		Pulse-induced incremental updates	Analog weight changes; low precision	100 μs to 100 ms	Mildly beneficial		Medium (sampling circuitry)
Reservoir computing & spatiotemporal detection	Neuron	Volatile memristor: VOx, WOx, NbOx, volatile ECM (e.g., Ag/SiO ₂)	Fading memory, nonlinear I–V, tunable τ	Binary or discretized multilevel; lower precision	10 μs to 100 ms	Beneficial		High (pulse timing, neuron circuit)
Stochastic computing—p-bit				Probabilistic switching	Binary; 1-bit precision	~100 ns	Beneficial	Medium (analog readout)
Hybrid static + dynamic (SNN, reservoir computing)	Synapse (static), Neuron (dynamic)	Volatile (static) + nonvolatile (dynamic)	Co-design of static MVM and transient dynamics	Multilevel (static), binary/fading (dynamic)	~10 ns (static), 10 μs to 100 ms (dynamic)	Harmful (static), beneficial (dynamic)	Low (static), High (dynamic)	Low (sampling circuitry)
								High (heterogeneous integration)

In this framework, Table 3 serves as the organizing framework of the discussion: it summarizes each application in terms of the typical device type, key computational feature, conductance states and precision, latency or characteristic time constants, role of randomness, endurance stress, and peripheral criticality.

Key device parameters determine suitability for specific computational primitives. For clarity, two operational modes are distinguished. Nonvolatile memristors, such as VCM, ECM, and PCM, provide persistent multilevel conductances and are the preferred substrate for dense MVM and many stateful or deterministic functions, including MVM, closed-loop AMC solvers, stateful logic, attractor networks, and STDP. Volatile memristors, such as WOx, NbOx, VOx, and part of Ag/Cu devices, exhibit fast decay and spontaneous reset, making them well suited for reservoir computing and spatiotemporal signal detection [13,16,30,31,171]. Stochastic computing generally follows two routes: continuous noisy weights realized with programmed nonvolatile memristors, or p-bit streams generated by volatile memristors. Furthermore, integrating both static and dynamic characteristics within hybrid architectures enables the realization of more complex applications [142]. Mapping any AI primitive onto hardware therefore begins with selecting device physics that match its functional role, namely persistence versus fading memory and deterministic versus stochastic behavior.

Table 3 further reveals that the same physical nonideality can play opposite roles depending on the application. Randomness is harmful for MVM and stateful logic when

accuracy and determinism are required, but it becomes beneficial for reservoir computing and stochastic computing, where the device variability itself provides the computational resource [5]. Likewise, the required number of conductance states differs sharply across applications: MVM and closed-loop AMC solver benefit from multilevel states, stateful logic is naturally binary or discretized multilevel, whereas reservoir computing and STDP tolerate lower precision. The latency or characteristic time constant also separates the application classes, from nanosecond-scale latency in MVM and stateful logic to microsecond-to-millisecond-scale dynamics in reservoir computing and STDP.

In this sense, Table 3 is not only a summary of device features, but also a practical map between physical behavior and algorithmic role. It captures why the same memristor-based platform can support dense analog computing, digital in-memory logic, temporal signal processing, and probabilistic sampling, while also clarifying which applications are dominated by precision, which are dominated by timing, and which are dominated by controlled stochasticity.

To contextualize the system-level benefits of these functional roles, Table 4 summarizes the quantitative performance advantages of memristor-based accelerators across various applications. Compared to conventional digital baselines, memristor-based implementations consistently demonstrate significant improvements in energy efficiency and latency. For instance, AMC architectures can deliver orders-of-magnitude energy savings for optimization tasks, while crossbar-based static conductance implementations drastically accelerate neural network inference and training. By centralizing these metrics, it becomes evident that while different applications exploit different device physics, they share the overarching architectural benefit of reducing data movement and leveraging massive parallelism.

Table 4. Performance summary of memristor-based computing systems across diverse applications.

Application	Problem	Array Size	Baseline Platform	Speedup	Energy Efficiency	Ref
MVM (CNN)	CIFAR-10	256 × 256	—	43 TOPS/W (peak)		[48]
AMC—linear regression	Signal processing	64 × 32	NVIDIA RTX 8000 GPU	—	1500×	[96]
AMC—Second-order Training	MNIST	256 × 256	NVIDIA V100 GPU	115.8×	41.9×	[99]
AMC—QP	Micro air vehicles control	100 × 100	STM32F405 MCU	2000×	5800×	[94]
AMC—PCA	Eigenface reconstruction	100 × 100	—	21 TFLOPS/W		[82]
Stateful logic (CNN)	ImageNet	1024 × 1024	NVIDIA GTX 1080 GPU	303.2×	297.9×	[128]
Attractor networks	associative memory	8 attractors	—	~100 ns	—	[137]
Reservoir computing	Gesture recognition	10 reservoirs	Intel i5-4460T CPU	Real-time	~1000×	[146]
STDP	Visual pattern learning	18 synapses	CMOS neuromorphic systems	~100×	—	[169]
Stochastic computing—continuous noisy weights	Robot's pouring skill learning	4K	NVIDIA A100 GPU	1.44×	153×	[175]
Stochastic computing—p-bit	Boolean logic	4K	CMOS ALU, Quantum computing	One-shot	Low	[171]

Note: “—” indicates data not reported or not applicable.

5.2. Practical Deployment Challenges and Mitigations

Although the application landscape is broad, practical deployment is still constrained by several interacting bottlenecks. For static conductance-based computing, accuracy and long-term stability are paramount [40]. Whether executing dense MVM or closed-loop AMC solvers, the effective numerical accuracy is limited by device nonlinearity, conductance variation, finite state resolution, and circuit noise [183,184]. High-precision analog computing explicitly treats reading noise and writing variability as primary obstacles, demonstrating that precision improvements require architectural co-design rather than device optimization alone. Recent work shows that arbitrarily high precision can be approached through carefully designed programming protocols and architecture-level error management, but at the cost of additional hardware and control complexity [35].

For MVM, the main practical response to this precision bottleneck has been bit slicing, mixed precision accumulation, and calibration-assisted mapping [34,48,185,186]. Bit slicing distributes one high-precision operation across multiple lower-precision arrays or multiple readout cycles and then recombines the partial results, which improves effective precision but inevitably increases array count, peripheral activity, and recombination cost [185]. The benefit is therefore not free precision, but a controlled precision energy trade off: more hardware and more readout steps obtain better accuracy, yet the system must still preserve the throughput advantage that motivated memristor-based MVM in the first place. This is especially important as array dimension grows, because wire resistance, voltage drop, and device mismatch all accumulate and amplify output error [48,183].

Closed-loop AMC circuit faces a different, and in some sense harder, precision problem. The closed-loop AMC solver is not only affected by the same conductance variability and interconnect resistance as MVM, but also by the stability of the feedback loop itself. Op-amp bandwidth, offset, settling dynamics, and loop gain all enter the accuracy equation, so the closed-loop AMC solver can become fragile long before the device array reaches its nominal size [79]. BlockAMC addresses the scalability barrier of large AMC by partitioning a large matrix into smaller block matrices that can be mapped to hardware-feasible array sizes, and by reconstructing the global solution from block-level solves [187]. To address the precision limitations of closed-loop AMC solvers, iterative refinement schemes leverage MVM to progressively minimize residual errors, combining an initial low-precision analog solution with high-precision MVM updates [188].

For dynamic switching applications, the dominant challenge is not only numerical precision, but also timing, endurance, and the controlled exploitation of stochasticity [189,190]. Stateful logic is a Boolean operation, so its scaling challenge is not correctness of the primitive itself, but the degree of parallel execution and the routing complexity needed to chain many gate instances. In practice, the output of stateful logic depends on the switching probability of the memristor and on the precise pulse conditions used to induce SET or RESET transitions. When many logic stages are cascaded, these probabilistic switching errors and the associated pulse propagation delay can accumulate, which increases the risk of incorrect outputs and reduces overall throughput. As a result, error correction or redundancy may be needed to maintain acceptable reliability. Reservoir computing scales differently. Its performance depends more on state richness, temporal diversity, and the matching between device time constants and task dynamics than on the raw size of the array. Experimental work shows that mask length and input scaling can tune reservoir richness, and that virtual node generation allows a small number of physical devices to emulate a much larger effective high-dimensional reservoir [145,191]. Thus, reservoir scalability is often improved through architectural expansion of the effective state space, rather than solely by increasing the number of physical devices.

Attractor networks, STDP, and stochastic computing have their own deployment limits. For attractor networks, the issue is not simply size, but the preservation of a stable energy landscape as the number of stored attractors increases. Once device mismatch and stochastic switching accumulate, the separation between attractor basins can blur, making recall less reliable [137]. For STDP, the challenge is precise spike timing, update symmetry, and reproducibility across many updates. For stochastic computing, the challenge is whether the switching statistics remain well calibrated over time so that probabilistic behavior stays meaningful rather than drifting into uncontrolled noise [167,171]. In these dynamic settings, the performance bottleneck is often the fidelity of local update statistics.

Across both modalities, manufacturing and commercialization impose constraints that extend beyond laboratory demonstrations. Practical memristor systems must be compatible with backend processing, foundry rules, and wafer-level yield requirements. Reported foundry-compatible crossbar circuits already demonstrate that low-temperature processing and high yield are possible, but they also make clear that variability control and device uniformity remain central to system-level accuracy [192]. More broadly, scalable deployment requires compact models, standardized test flows, and calibration procedures that can be used in a production design environment [193]. Without these ingredients, even impressive device demonstrations remain difficult to translate into reliable products. In this respect, commercialization is not only a materials issue, but also a software and verification issue, because the system must be predictable enough to be deployed repeatedly rather than just characterized once in a laboratory.

5.3. Future Research Directions

Future research should follow a full stack co-design path. At the device level, the field still needs improved control over linearity, symmetry, endurance, and conductance uniformity to support high-precision static workloads, alongside tunable relaxation kinetics and calibrated stochasticity for dynamic applications [48,170]. At the circuit level, more robust selectors, compensation schemes, and peripheral reduction strategies are needed to alleviate IR drop and ADC or DAC overhead. At the architecture level, the focus should shift toward heterogeneous integration, where static conductance arrays are deployed for deterministic linear algebra problem, dynamic switching substrates handle temporal and stochastic computations, and digital controllers manage orchestration, precision scaling, and error resilience [142]. At the algorithm level, hardware aware training, noise-aware optimization, and mixed-precision mapping should be developed together with device constraints in mind rather than as post hoc adaptations [89,193].

Complementing these hardware and algorithmic advances, software support is another important direction that has not yet been explored deeply enough. A realistic memristor-based computing platform will require robust compilers and toolchains that can map neural networks or optimization problems onto array tiles, schedule peripheral conversions, and manage precision decomposition automatically. Such software must also expose the hardware constraints to the user-level training framework, so that models can be adapted before deployment rather than after failure. This is especially important for LLMs and transformer-based systems, where hardware-aware training, blocked attention, and memory efficient key value handling are likely to be more effective than a direct mapping of full precision operations onto crossbar arrays [194]. In practice, the challenge is not merely accelerating individual computational primitives, but reducing end-to-end latency and data movement under realistic memory and bandwidth limits. Ultimately, establishing standardized compiler toolchains, robust calibration strategies, and reproducible system benchmarks will be essential before memristor-based accelerators can transition from laboratory prototypes to robust, deployable technologies.

Finally, deployment scenarios should be made more explicit. Edge systems, near sensor processing, and embedded neuromorphic platforms emphasize low-power, low-latency, and event-driven operation [42]. By contrast, larger scale accelerators emphasize throughput, bandwidth, and predictable integration with existing software stacks. A mature memristor-based AI ecosystem will therefore need to support both kinds of deployment, while matching hardware modalities to task structure. The most promising future direction is not a universal memristor platform for all applications, but a family of co-designed solutions in which device physics, circuit topology, algorithmic tolerance, compiler support, and deployment context are aligned for a specific class of workloads.

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 MVM and supports closed-loop AMC solvers. Together, these primitives directly accelerate core linear-algebra workloads in AI, spanning NN inference and training, as well as regression, classification, and constrained optimization tasks. Dynamic switching modality—spanning volatile relaxation, nonvolatile stateful transitions, and intrinsic stochasticity—supply fading memory, nonlinear activation, and probabilistic state transitions, enabling hardware-native implementations of stateful logic, attractor networks, reservoir computing, STDP, and Bayesian sampling. Together, these 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 validate both static and dynamic paradigms. For instance, static crossbars have been shown to accelerate NN inference and one-step matrix equation solving, whereas dynamic switching substrates enable real-time reservoir computing, online STDP learning, and probabilistic Bayesian sampling. 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 the full potential of practical memristor-based AI requires integrated solutions. We advocate coordinated device–circuit–algorithm co-design, task-specific heterogeneous architectures that strategically assign each modality to its most effective, robust calibration and pulse-engineering strategies, and standardized software toolchains for automated workload mapping. Standardized benchmarks and reproducible system demonstrations will be essential to quantify benefits. With these efforts, memristor technologies can move from compelling laboratory demonstrations to scalable, application-ready accelerators, fundamentally reshaping the landscape of next-generation, hardware-native AI.

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. (Yongxiang Li), Y.L. (Yubiao Luo), J.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: The original contributions presented in the study are included in the article. Further inquiries can be directed to the corresponding author.

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

Appendix A

To facilitate reading and ensure clarity amidst the mathematical formulations, neural network models, and physical device dynamics discussed in this review, Table A1 provides a comprehensive summary of the principal symbols and their definitions. Furthermore, this table explicitly defines specific symbols that assume context-dependent meanings across different sections of the manuscript.

Table A1. Symbol notation.

Symbol	Meaning	Symbol	Meaning
A	Weight matrix	C	Covariance matrix
G	Conductance matrix	D	Graph degree matrix
x	MVM input/AMC output vector	I	Identity matrix
y	MVM output/AMC input vector	g_x	Input conductance vector
v	Output voltage vector	k	Constant of dual IMC
i	Input current vector	g_c	Compensation conductance
z	Intermediate vector	μ	Auxiliary node voltage
W	Weight matrix of NN	λ	Eigenvalue parameter
b	Bias vector of the NN	τ	Relaxation time
H	Node-feature matrix of GNN	Δt	Relative spike timing
Q	Query matrices	m, n	Matrix dimensions
K	Key matrix	V	Node voltage
V	Value matrix	I	Current
L	Graph Laplacian matrix	g	Memristor conductance
h	hidden state of RNN	β	Sparsity penalty

References

- Horowitz, M. Computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*; IEEE: San Francisco, CA, USA, 2014; pp. 10–14.
- Waldrop, M.M. The chips are down for Moore's law. *Nature* **2016**, *530*, 144–147. [[CrossRef](#)] [[PubMed](#)]
- Ielmini, D.; Wong, H.-S.P. In-memory computing with resistive switching devices. *Nat. Electron.* **2018**, *1*, 333–343. [[CrossRef](#)]
- Sun, Z.; Kvatinsky, S.; Si, X.; Mehonic, A.; Cai, Y.; Huang, R. A full spectrum of computing-in-memory technologies. *Nat. Electron.* **2023**, *6*, 823–835. [[CrossRef](#)]
- Sebastian, A.; Le Gallo, M.; Khaddam-Aljameh, R.; Eleftheriou, E. Memory devices and applications for in-memory computing. *Nat. Nanotechnol.* **2020**, *15*, 529–544. [[CrossRef](#)]
- 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. [[CrossRef](#)]
- Ielmini, D. Resistive switching memories based on metal oxides: Mechanisms, reliability and scaling. *Semicond. Sci. Technol.* **2016**, *31*, 063002. [[CrossRef](#)]
- Mehonic, A.; Ielmini, D.; Roy, K.; Mutlu, O.; Kvatinsky, 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. [[CrossRef](#)]
- Mannocci, P.; Farronato, M.; Lepri, N.; Cattaneo, L.; Glukhov, A.; Sun, Z.; Ielmini, D. In-Memory Computing with Emerging Memory Devices: Status and Outlook. *APL Mach. Learn.* **2023**, *1*, 010902. [[CrossRef](#)]
- 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. [[CrossRef](#)] [[PubMed](#)]
- 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. USA* **2019**, *116*, 4123–4128. [[CrossRef](#)]
- Amirsoleimani, A.; Alibart, F.; Yon, V.; Xu, J.; Pazhouhandeh, M.R.; Ecoffey, S.; Beilliard, Y.; Genov, R.; Drouin, D. In-Memory Vector-Matrix Multiplication in Monolithic Complementary Metal–Oxide–Semiconductor-Memristor Integrated Circuits: Design Choices, Challenges, and Perspectives. *Adv. Intell. Syst.* **2020**, *2*, 2000115. [[CrossRef](#)]
- 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. [[CrossRef](#)] [[PubMed](#)]

14. 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. [\[CrossRef\]](#)
15. 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. [\[CrossRef\]](#)
16. 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. [\[CrossRef\]](#)
17. 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. [\[CrossRef\]](#)
18. Chua, L.O. Memristor—The missing circuit element. *IEEE Trans. Circuit Theory* **1971**, *18*, 507–519. [\[CrossRef\]](#)
19. Strukov, D.B.; Snider, G.S.; Stewart, D.R.; Williams, R.S. The missing memristor found. *Nature* **2008**, *453*, 80–83. [\[CrossRef\]](#)
20. 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. [\[CrossRef\]](#) [\[PubMed\]](#)
21. 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. [\[CrossRef\]](#)
22. 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. [\[CrossRef\]](#)
23. 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. [\[CrossRef\]](#)
24. Valov, I.; Waser, R.; Jameson, J.R.; Kozicki, M.N. Electrochemical metallization memories—Fundamentals, applications, prospects. *Nanotechnology* **2011**, *22*, 254003. [\[CrossRef\]](#)
25. 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. [\[CrossRef\]](#)
26. Raoux, S.; Welnic, W.; Ielmini, D. Phase change materials and their application to nonvolatile memories. *Chem. Rev.* **2010**, *110*, 240–267. [\[CrossRef\]](#) [\[PubMed\]](#)
27. Wuttig, M.; Yamada, N. Phase-change materials for rewriteable data storage. *Nat. Mater.* **2007**, *6*, 824–832. [\[CrossRef\]](#) [\[PubMed\]](#)
28. Nandakumar, S.R.; Le Gallo, M.; Boybat, I.; Rajendran, B.; Sebastian, A.; Eleftheriou, E. A Phase-Change Memory Model for Neuromorphic Computing. *J. Appl. Phys.* **2018**, *124*, 152135. [\[CrossRef\]](#)
29. 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. [\[CrossRef\]](#) [\[PubMed\]](#)
30. 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. [\[CrossRef\]](#)
31. Ju, D.; Kim, S. Versatile NbO_x-based volatile memristor for artificial intelligent applications. *Adv. Funct. Mater.* **2024**, *34*, 2409436. [\[CrossRef\]](#)
32. Chekol, S.A.; Nacke, R.; Aussen, S.; Hoffmann-Eifert, S. SET Kinetics of Ag/HfO₂-Based Diffusive Memristors under Various Counter-Electrode Materials. *Micromachines* **2023**, *14*, 571. [\[CrossRef\]](#)
33. 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. [\[CrossRef\]](#) [\[PubMed\]](#)
34. 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. [\[CrossRef\]](#)
35. 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. [\[CrossRef\]](#)
36. 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. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*; IEEE: Macao, China, 2016; pp. 280–285.
37. Sun, Z.; Huang, R. Time complexity of in-memory matrix-vector multiplication. *IEEE Trans. Circuits Syst. II Express Briefs* **2021**, *68*, 2785–2789. [\[CrossRef\]](#)
38. Antolini, A.; Lico, A.; Zavalloni, F.; Greco, L.; Zurla, R.; Bertolini, J.; Vignali, R.; Iannelli, L.; Calvetti, E.; Pasotti, M.; et al. High-Precision Close-to-Analog Programming of PCM Cells as Devices for AiMC Edge-AI. In *Proceedings IEEE European Solid-State Electronics Research Conference (ESSERC)*; IEEE: Munich, Germany, 2025.
39. Wang, S.; Sun, Z. Dual in-memory computing of matrix-vector multiplication for accelerating neural networks. *Device* **2024**, *2*, 100546. [\[CrossRef\]](#)

40. 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. [\[CrossRef\]](#)
41. 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. [\[CrossRef\]](#)
42. 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. In *2018 IEEE International Solid-State Circuits Conference (ISSCC)*; IEEE: San Francisco, CA, USA, 2018; pp. 494–496.
43. Gokmen, T.; Vlasov, Y. Acceleration of deep neural network training with resistive cross-point devices: Design considerations. *Front. Neurosci.* **2016**, *10*, 333. [\[CrossRef\]](#)
44. 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. [\[CrossRef\]](#)
45. 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. [\[CrossRef\]](#)
46. 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. In *Proceedings of the 53rd Annual Design Automation Conference (DAC)*; Association for Computing Machinery: New York, NY, USA, 2016; pp. 1–6.
47. Lecun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [\[CrossRef\]](#)
48. 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. [\[CrossRef\]](#)
49. 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. [\[CrossRef\]](#)
50. 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. [\[CrossRef\]](#)
51. Yakopcic, C.; Alom, M.Z.; Taha, T.M. Memristor crossbar deep network implementation based on a convolutional neural network. In *2016 International Joint Conference on Neural Networks (IJCNN)*; IEEE: San Francisco, CA, USA, 2016; pp. 963–970.
52. 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. [\[CrossRef\]](#)
53. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [\[CrossRef\]](#)
54. 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. [\[CrossRef\]](#)
55. 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. [\[CrossRef\]](#)
56. Liu, X.; Zeng, Z.; Wunsch, D.C., II. Memristor-based LSTM network with in situ training and its applications. *Neural Netw.* **2020**, *131*, 300–311. [\[CrossRef\]](#)
57. Smagulova, K.; James, A.P. A survey on LSTM memristive neural network architectures and applications. *Eur. Phys. J. Spec. Top.* **2019**, *228*, 2313–2324. [\[CrossRef\]](#)
58. Nikam, H.; Satyam, S.; Sahay, S. Long short-term memory implementation exploiting passive RRAM crossbar array. *IEEE Trans. Electron Devices* **2021**, *69*, 1743–1751. [\[CrossRef\]](#)
59. Koller, D.; Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*; MIT Press: Cambridge, MA, USA, 2009.
60. 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. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*; IEEE: San Francisco, CA, USA, 2021; pp. 1667–1672.
61. 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. [\[CrossRef\]](#)
62. Morsali, M.; Nazzal, M.; Khreishah, A.; Angizi, S. Ima-GNN: In-memory acceleration of centralized and decentralized graph neural networks at the edge. In *Proceedings of the Great Lakes Symposium on VLSI 2023*; Association for Computing Machinery: New York, NY, USA, 2023; pp. 3–8.
63. 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. In *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC)*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 469–474.
64. 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. Comput.-Aided Des. Integr. Circuits Syst.* **2024**, *43*, 2635–2646. [\[CrossRef\]](#)
65. 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. Comput.-Aided Des. Integr. Circuits Syst.* **2022**, *42*, 2703–2716. [\[CrossRef\]](#)

66. 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 Unconv. Comput.* **2024**, *1*, 6. [\[CrossRef\]](#)
67. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2017.
68. Yang, X.; Yan, B.; Li, H.; Chen, Y. ReTransformer: ReRAM-based processing-in-memory architecture for transformer acceleration. In *Proceedings of the 39th International Conference on Computer-Aided Design (ICCAD)*; Association for Computing Machinery: New York, NY, USA, 2020; pp. 1–9.
69. 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. [\[CrossRef\]](#)
70. Liu, A.; Liu, J.; Pan, Z.; He, Y.; Haffari, G.; Zhuang, B. MiniCache: KV Cache Compression in Depth Dimension for Large Language Models. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 139997–140031.
71. 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. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*; Association for Computing Machinery: New York, NY, USA, 2024; pp. 103–119.
72. Zhou, M.; Xu, W.; Kang, J.; Rosing, T. TransPIM: A memory-based acceleration via software-hardware co-design for transformer. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*; IEEE: San Francisco, CA, USA, 2022; pp. 1071–1085.
73. Liu, S.; Mu, C.; Jiang, H.; Wang, Y.; Zhang, J.; Lin, F.; Zhou, K.; Liu, Q.; Chen, C. HARDSEA: Hybrid Analog-ReRAM Clustering and Digital-SRAM In-Memory Computing Accelerator for Dynamic Sparse Self-Attention in Transformer. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2024**, *32*, 269–282. [\[CrossRef\]](#)
74. Bettayeb, M.; Halawani, Y.; Khan, M.U.; Saleh, H.; Mohammad, B. Efficient memristor accelerator for transformer self-attention functionality. *Sci. Rep.* **2024**, *14*, 24173. [\[CrossRef\]](#)
75. Yang, C.; Wang, X.; Zeng, Z. Full-circuit implementation of transformer network based on memristor. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2022**, *69*, 1395–1407. [\[CrossRef\]](#)
76. 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. [\[CrossRef\]](#)
77. Nowshin, F.; Yi, Y. Memristor-based deep spiking neural network with a computing-in-memory architecture. In *2022 23rd International Symposium on Quality Electronic Design (ISQED)*; IEEE: San Francisco, CA, USA, 2022; pp. 1–6.
78. 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. [\[CrossRef\]](#)
79. 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. [\[CrossRef\]](#)
80. Sun, Z.; Pedretti, G.; Ambrosi, E.; Bricalli, A.; Ielmini, D. In-Memory Eigenvector Computation in Time $O(1)$. *Adv. Intell. Syst.* **2020**, *2*, 2000042. [\[CrossRef\]](#)
81. 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. [\[CrossRef\]](#)
82. 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. Pap.* **2025**; early access.
83. 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. [\[CrossRef\]](#) [\[PubMed\]](#)
84. Kutner, M.H.; Nachtsheim, C.J.; Neter, J.; Li, W. *Applied Linear Statistical Models*, 5th ed.; McGraw-Hill: New York, NY, USA, 2004.
85. Sun, Z.; Pedretti, G.; Bricalli, A.; Ielmini, D. One-step regression and classification with cross-point resistive memory arrays. *Sci. Adv.* **2020**, *6*, eaay2378. [\[CrossRef\]](#) [\[PubMed\]](#)
86. 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. [\[CrossRef\]](#)
87. Farronato, M.; Mannocci, P.; Melegari, M.; Ricci, S.; Compagnoni, C.M.; Ielmini, D. Reservoir computing with charge-trap memory based on a MoS₂ channel for neuromorphic engineering. *Adv. Mater.* **2023**, *35*, 2205381. [\[CrossRef\]](#)
88. Mikolajick, T.; Park, M.H.; Begon-Lours, L.; Slesazek, S. From ferroelectric material optimization to neuromorphic devices. *Adv. Mater.* **2023**, *35*, 2206042. [\[CrossRef\]](#) [\[PubMed\]](#)
89. 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. [\[CrossRef\]](#)

90. 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. [\[CrossRef\]](#)
91. 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. [\[CrossRef\]](#)
92. Gómez-Luna, J.; Guo, Y.; Brocard, S.; Legriel, J.; Cimadomo, R.; Oliveira, G.F. Evaluating machine learning workloads on memory-centric computing systems. In *2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*; IEEE: San Francisco, CA, USA, 2023; pp. 35–49.
93. 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. Pap.* **2021**, *68*, 4900–4909. [\[CrossRef\]](#)
94. 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. [\[CrossRef\]](#)
95. 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. Pap.* **2021**, *68*, 4889–4899. [\[CrossRef\]](#)
96. 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. [\[CrossRef\]](#)
97. 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. [\[CrossRef\]](#)
98. 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. [\[CrossRef\]](#)
99. Zhao, Y.; Liu, F.; Gao, M.; Liang, X.; Tang, Q.; Gu, C.; Yang, T.; Jing, N.; Jiang, L. STAMP: Accelerating Second-Order DNN Training Via ReRAM-Based Processing-in-Memory Architecture. In *International Symposium on Advanced Parallel Processing Technologies*; Springer Nature: Singapore, 2025; pp. 160–170.
100. Song, L.; Qian, X.; Li, H.; Chen, Y. PipeLayer: A Pipelined ReRAM-Based Accelerator for Deep Learning. In *Proceedings IEEE International Symposium on High Performance Computer Architecture (HPCA)*; IEEE: San Francisco, CA, USA, 2017; pp. 541–552.
101. 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. [\[CrossRef\]](#) [\[PubMed\]](#)
102. Aifer, M.; Donatella, K.; Gordon, M.H.; Duffield, S.; Ahle, T.; Simpson, D.; Crooks, G.; Coles, P.J. Thermodynamic linear algebra. *npj Unconv. Comput.* **2024**, *1*, 13. [\[CrossRef\]](#)
103. 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. In *Proceedings of the NeurIPS 2023 Workshop on Machine Learning with New Compute Paradigms*, New Orleans, LA, USA, 10–16 December 2023.
104. Aifer, M.; Duffield, S.; Donatella, K.; Melanson, D.; Klett, P.; Belateche, Z.; Crooks, G.; Martinez, A.J.; Coles, P.J. Thermodynamic bayesian inference. In *2024 IEEE International Conference on Rebooting Computing (ICRC)*; IEEE: San Francisco, CA, USA, 2024; pp. 1–13.
105. Donatella, K.; Duffield, S.; Aifer, M.; Melanson, D.; Crooks, G.; Coles, P.J. Thermodynamic natural gradient descent. *npj Unconv. Comput.* **2026**, *3*, 5. [\[CrossRef\]](#)
106. 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* **2025**, arXiv:2502.08603. [\[CrossRef\]](#)
107. Boyd, S.; Vandenberghe, L. *Convex Optimization*; Cambridge University Press: Cambridge, UK, 2004.
108. Wu, L.; Braatz, R.D. A Quadratic Programming Algorithm with $O(n^3)$ Time Complexity. In *IEEE Transactions on Automatic Control*; IEEE: San Francisco, CA, USA, 2026; pp. 1–16.
109. Cohen, M.B.; Lee, Y.T.; Song, Z. Solving Linear Programs in the Current Matrix Multiplication Time. *J. ACM* **2021**, *68*, 1–39. [\[CrossRef\]](#)
110. 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. [\[CrossRef\]](#)
111. 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. [\[CrossRef\]](#)
112. Gao, X.; Liao, L.Z. A New One-Layer Neural Network for Linear and Quadratic Programming. *IEEE Trans. Neural Netw.* **2010**, *21*, 918–929. [\[PubMed\]](#)
113. 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. [\[CrossRef\]](#)
114. Vichik, S.; Borrelli, F. Solving linear and quadratic programs with an analog circuit. *Comput. Chem. Eng.* **2014**, *70*, 160–171. [\[CrossRef\]](#)

115. Vichik, S.; Arcak, M.; Borrelli, F. Stability of an analog optimization circuit for quadratic programming. *Syst. Control Lett.* **2016**, *88*, 68–74. [\[CrossRef\]](#)
116. 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. [\[CrossRef\]](#)
117. Von Luxburg, U. A Tutorial on Spectral Clustering. *Stat. Comput.* **2007**, *17*, 395–416. [\[CrossRef\]](#)
118. Chi, Y.; Song, X.; Zhou, D.; Hino, K.; Tseng, B.L. Evolutionary Spectral Clustering by Incorporating Temporal Smoothness. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*; Association for Computing Machinery: New York, NY, USA, 2007; pp. 153–162.
119. Leskovec, J.; Lang, K.J.; Mahoney, M. Empirical Comparison of Algorithms for Network Community Detection. In *Proceedings of the 19th International Conference on World Wide Web*; Association for Computing Machinery: New York, NY, USA, 2010; pp. 631–640.
120. 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. [\[CrossRef\]](#)
121. 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. [\[CrossRef\]](#)
122. Kvatsinsky, 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. [\[CrossRef\]](#)
123. Sun, Z.; Ambrosi, E.; Bricalli, A.; Ielmini, D. Logic Computing with Stateful Neural Networks of Resistive Switches. *Adv. Mater.* **2018**, *30*, 1802554. [\[CrossRef\]](#)
124. 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. [\[CrossRef\]](#)
125. 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. [\[CrossRef\]](#)
126. 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. [\[CrossRef\]](#)
127. Ben-Hur, R.; Ronen, R.; Haj-Ali, A.; Bhattacharjee, D.; Eliahu, A.; Peled, N.; Kvatsinsky, S. SIMPLER MAGIC: Synthesis and Mapping of In-Memory Logic Executed in a Single Row to Improve Throughput. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2019**, *39*, 2434–2447. [\[CrossRef\]](#)
128. Imani, M.; Gupta, S.; Kim, Y.; Rosing, T. FloatPIM: In-Memory Acceleration of Deep Neural Network Training with High Precision. In *Proceedings of the 46th International Symposium on Computer Architecture (ISCA)*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 802–815.
129. 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. [\[CrossRef\]](#)
130. Imani, M.; Gupta, S.; Rosing, T. Digital-Based Processing in-Memory: A Highly-Parallel Accelerator for Data Intensive Applications. In *Proceedings of the International Symposium on Memory Systems*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 38–40.
131. Leitersdorf, O.; Ronen, R.; Kvatsinsky, S. Making Memristive Processing-in-Memory Reliable. In *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*; IEEE: San Francisco, CA, USA, 2021; pp. 1–6.
132. Leitersdorf, O.; Ronen, R.; Kvatsinsky, S. ConvPIM: Evaluating Digital Processing-in-Memory through Convolutional Neural Network Acceleration. *arXiv* **2025**, arXiv:2305.04122.
133. Lu, Z.; Arafat, M.T.; Qu, G. RIME: A Scalable and Energy-Efficient Processing-in-Memory Architecture for Floating-Point Operations. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*; Association for Computing Machinery: New York, NY, USA, 2021; pp. 120–125.
134. 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. [\[CrossRef\]](#)
135. Imani, M.; Pampana, S.; Gupta, S.; Zhou, M.; Kim, Y.; Rosing, T. DUAL: Acceleration of Clustering Algorithms Using Digital-Based Processing in-Memory. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*; IEEE: San Francisco, CA, USA, 2020; pp. 356–371.
136. Rashed, M.R.H.; Jha, S.K.; Ewetz, R. Hybrid Analog-Digital In-Memory Computing. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*; IEEE: San Francisco, CA, USA, 2021; pp. 1–9.
137. 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. [\[CrossRef\]](#)
138. 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. [\[CrossRef\]](#)

139. 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. [\[CrossRef\]](#)
140. 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. [\[CrossRef\]](#)
141. Liang, X.; Tang, J.; Zhong, Y.; Gao, B.; Qian, H.; Wu, H. Physical Reservoir Computing with Emerging Electronics. *Nat. Electron.* **2024**, *7*, 193–206. [\[CrossRef\]](#)
142. 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. [\[CrossRef\]](#)
143. 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. [\[CrossRef\]](#)
144. 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. [\[CrossRef\]](#)
145. 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. [\[CrossRef\]](#) [\[PubMed\]](#)
146. 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. [\[CrossRef\]](#)
147. Wang, S.; Gao, S.; Tang, C.; Occhipinti, E.; Li, C.; Wang, S.; Wang, J.; Zhao, H.; Hu, G.; Nathan, A.; et al. Memristor-Based Adaptive Neuromorphic Perception in Unstructured Environments. *Nat. Commun.* **2024**, *15*, 4671. [\[CrossRef\]](#)
148. 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. [\[CrossRef\]](#)
149. 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. [\[CrossRef\]](#)
150. 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. [\[CrossRef\]](#)
151. 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. [\[CrossRef\]](#) [\[PubMed\]](#)
152. 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. [\[CrossRef\]](#)
153. Milozzi, A.; Ricci, S.; Ielmini, D. Memristive Tonotopic Mapping with Volatile Resistive Switching Memory Devices. *Nat. Commun.* **2024**, *15*, 2812. [\[CrossRef\]](#) [\[PubMed\]](#)
154. 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. [\[CrossRef\]](#)
155. 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. [\[CrossRef\]](#)
156. Roy, K.; Jaiswal, A.; Panda, P. Towards Spike-Based Machine Intelligence with Neuromorphic Computing. *Nature* **2019**, *575*, 607–617. [\[CrossRef\]](#) [\[PubMed\]](#)
157. 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. [\[CrossRef\]](#)
158. 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. [\[CrossRef\]](#)
159. 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. [\[CrossRef\]](#)
160. 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. [\[CrossRef\]](#)
161. 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. [\[CrossRef\]](#)
162. Indiveri, G.; Linares-Barranco, B.; Legenstein, R.; Deligeorgis, G.; Prodromakis, T. Integration of Nanoscale Memristor Synapses in Neuromorphic Computing Architectures. *Nanotechnology* **2013**, *24*, 384010. [\[CrossRef\]](#)
163. 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. [\[CrossRef\]](#)

164. 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. [\[CrossRef\]](#) [\[PubMed\]](#)
165. 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. [\[CrossRef\]](#) [\[PubMed\]](#)
166. 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. [\[CrossRef\]](#)
167. 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. [\[CrossRef\]](#) [\[PubMed\]](#)
168. 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. [\[CrossRef\]](#)
169. 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. [\[CrossRef\]](#) [\[PubMed\]](#)
170. Ding, C.; Ren, Y.; Liu, Z.; Wong, N. Transforming Memristor Noises into Computational Innovations. *Commun. Mater.* **2025**, *6*, 149. [\[CrossRef\]](#)
171. Woo, K.S.; Kim, J.; Han, J.; Kim, W.; Jang, Y.H.; Hwang, C.S. Probabilistic Computing Using Cu_{0.1}Te_{0.9}/HfO₂/Pt Diffusive Memristors. *Nat. Commun.* **2022**, *13*, 5762. [\[CrossRef\]](#)
172. Wang, K.; Hu, Q.; Gao, B.; Lin, Q.; Zhuge, F.; Zhang, D.; Wang, L.; He, Y.; Scheicher, R.H.; Tong, H.; et al. Threshold Switching Memristor-Based Stochastic Neurons for Probabilistic Computing. *Mater. Horiz.* **2021**, *8*, 619–629. [\[CrossRef\]](#)
173. 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. [\[CrossRef\]](#)
174. 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. [\[CrossRef\]](#)
175. 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. [\[CrossRef\]](#)
176. 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. [\[CrossRef\]](#) [\[PubMed\]](#)
177. 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. [\[CrossRef\]](#)
178. 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. [\[CrossRef\]](#)
179. Gao, D.; Huang, Q.; Zhang, G.L.; Yin, X.; Li, B.; Schlichtmann, U.; Zhuo, C. Bayesian Inference Based Robust Computing on Memristor Crossbar. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*; IEEE: San Francisco, CA, USA, 2021; pp. 121–126.
180. 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. In *2019 IEEE International Electron Devices Meeting (IEDM)*; IEEE: San Francisco, CA, USA, 2019; pp. 14.6.1–14.6.4.
181. Bonnet, D.; Hirtzlin, T.; Majumdar, A.; Dalgaty, T.; Esmanhotto, 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. [\[CrossRef\]](#) [\[PubMed\]](#)
182. 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. Comput.-Aided Des. Integr. Circuits Syst.* **2022**, *42*, 2136–2148. [\[CrossRef\]](#)
183. 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. [\[CrossRef\]](#)
184. Zolfagharinejad, 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. [\[CrossRef\]](#)
185. Feinberg, B.; Vengalam, U.K.R.; Whitehair, N.; Wang, S.; Ipek, E. Enabling Scientific Computing on Memristive Accelerators. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*; IEEE: San Francisco, CA, USA, 2018; pp. 367–382.
186. 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. [\[CrossRef\]](#)

187. Pan, L.; Zuo, P.; Luo, Y.; Sun, Z.; Huang, R. BlockAMC: Scalable In-Memory Analog Matrix Computing for Solving Linear Systems. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*; IEEE: San Francisco, CA, USA, 2024; pp. 1–6.
188. Zuo, P.; Wang, Q.; Luo, Y.; Xie, R.; Wang, S.; Cheng, Z.; Bao, L.; Wang, Z.; Cai, Y.; Huang, R.; et al. Precise and Scalable Analogue Matrix Equation Solving Using Resistive Random-Access Memory Chips. *Nat. Electron.* **2025**, *8*, 1222–1233. [[CrossRef](#)]
189. 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. [[CrossRef](#)]
190. 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. [[CrossRef](#)]
191. 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. [[CrossRef](#)]
192. Choi, S.; Bezugam, S.S.; Bhattacharya, T.; Kwon, D.; Strukov, D.B. Wafer-Scale Fabrication of Memristive Passive Crossbar Circuits for Brain-Scale Neuromorphic Computing. *Nat. Commun.* **2025**, *16*, 8757. [[CrossRef](#)]
193. Rasch, M.J.; Mackin, C.; Le Gallo, M.; Chen, A.; Fasoli, A.; Odermatt, F.; Li, N.; Nandakumar, S.R.; Narayanan, P.; Tsai, H.; et al. Hardware-Aware Training for Large-Scale and Diverse Deep Learning Inference Workloads Using In-Memory Computing-Based Accelerators. *Nat. Commun.* **2023**, *14*, 5282. [[CrossRef](#)]
194. Spoon, K.; Tsai, H.; Chen, A.; Rasch, M.J.; Ambrogio, S.; Mackin, C.; Fasoli, A.; Friz, A.M.; Narayanan, P.; Stanisavljevic, M.; et al. Toward Software-Equivalent Accuracy on Transformer-Based Deep Neural Networks with Analog Memory Devices. *Front. Comput. Neurosci.* **2021**, *15*, 675741. [[CrossRef](#)]

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.