

Memristor-Based In-Memory Computing Architecture for Energy-Efficient Deep Neural Network Acceleration

Dr. Elena Petrova¹, Prof. Michael O'Connor²

Associate Professor of Computer Science, Sofia Center for Intelligent Computing, Bulgaria.
Prof. Michael O'Connor, Professor of Electronic Engineering, Dublin Institute of Technology
and AI, Ireland.

Email: elena.petrova@scic-research.org, michael.oconnor@ditai.org

Article Received 11-07-2026, Revised 18-08-2026, Accepted 14-09-2026

Author(s) Retains the Copyrights of This Article

Abstract

The rapid expansion of deep neural networks has created a growing demand for computing platforms capable of delivering high inference performance within increasingly restricted energy and memory budgets. Conventional computing systems separate storage and computation, requiring model parameters and intermediate activation data to travel repeatedly between memory and processing units. This movement introduces latency, consumes additional energy, and becomes particularly restrictive when neural networks are deployed in edge and embedded environments. This research presents a memristor-based in-memory computing architecture intended to reduce this overhead by combining non-volatile weight storage with massively parallel computation. The proposed architecture employs memristive crossbar arrays in which neural-network parameters are represented by programmable conductance states. Input activations are applied to the array, while the resulting electrical responses are accumulated to perform vector-matrix multiplication within the memory structure. Peripheral conversion circuits, digital control logic, programmable weight mapping, precision management, and error-compensation mechanisms are incorporated to form a complete accelerator architecture. The research further considers device variability, conductance nonlinearity, limited precision, analog noise, and peripheral conversion overhead because these factors strongly influence the practical benefits of analog in-memory computing. A hardware-algorithm co-design methodology is introduced to determine suitable crossbar dimensions, parameter mapping strategies, numerical precision, and compensation methods for representative deep-learning workloads. The proposed framework is evaluated using inference accuracy, latency, energy per inference, throughput, memory utilization, and energy-delay product as principal performance indicators. The resulting analysis indicates that integrating storage and computation can substantially reduce the frequency of data transfers between conventional memory and processing elements. The architecture is therefore particularly relevant to edge artificial intelligence, embedded machine learning, intelligent sensors, Internet-of-Things systems, and other applications in which computational efficiency and energy consumption are important design constraints.

Keywords: *Memristor, In-Memory Computing, Processing-in-Memory, Deep Neural Networks, DNN Accelerator, Resistive RAM, ReRAM, Crossbar Array, Neuromorphic Computing, Energy-Efficient Computing, Vector-Matrix Multiplication, Hardware Acceleration, Artificial Intelligence, Non-Volatile Memory, Hardware-Algorithm Co-Design.*

1. Introduction

Deep neural networks have become an important computational foundation for image recognition, speech processing, natural-language applications, intelligent sensing, autonomous systems, and many other artificial-intelligence workloads. Their effectiveness is largely associated with the ability to learn high-dimensional representations through repeated matrix operations and nonlinear

transformations. As network depth, parameter count, and activation complexity increase, however, the computational platform must move a continuously growing volume of information between memory and processing resources. The resulting communication overhead can become a significant component of total execution time and energy consumption. The traditional von Neumann organization is particularly affected because memory and arithmetic operations

are physically separated, meaning that even relatively simple neural-network calculations can involve numerous memory-access operations. Memristor-based computing has therefore attracted attention because a memory element can simultaneously retain a parameter and participate in the computation associated with that parameter [1], [3], [8], [12].

Memristive devices are especially attractive for neural acceleration because their conductance can represent model parameters while their electrical behavior can participate directly in multiplication and accumulation. In a crossbar organization, a large number of programmable devices can be arranged at the intersections of conductive lines. Applying input values to one dimension of the array produces currents determined by the programmed conductance values. The aggregate current can therefore represent the result of a matrix–vector operation. This operating principle has been demonstrated through several generations of memristive and resistive-memory computing systems [4], [5], [13], [14]. The basic concept is consistent with the broader development of neuromorphic hardware, where memory, communication, and computation are increasingly treated as an integrated system rather than independent functional blocks.

The significance of the approach is not restricted to reducing memory traffic. A conventional digital accelerator generally performs multiplication and accumulation using arithmetic units, while weights are retrieved from SRAM, DRAM, or another storage hierarchy. The same parameters may be transferred repeatedly as different portions of an input tensor are processed. In contrast, an in-memory architecture attempts to keep the parameters physically close to the computational operation. This arrangement can reduce the number of explicit data-transfer events and can exploit the parallel electrical behavior of the crossbar. Studies of non-volatile-memory computing have consequently investigated both device-level properties and system-level architectures for improving energy efficiency and computational density [3], [12], [17], [20].

However, memristor-based computing is not simply a matter of replacing digital memory with resistive devices. Real devices exhibit conductance variation, programming errors, nonlinear switching, finite resistance ranges, retention effects, temperature dependence, and noise. The crossbar itself can also experience parasitic resistance, line losses, sneak-current effects, and limited signal margins. Peripheral circuits such as digital-to-analog converters, analog-to-digital converters, sensing circuits, drivers, and control logic can consume a substantial fraction of the

system energy if they are not carefully designed. Consequently, a useful accelerator must be considered across multiple abstraction levels. The importance of integrated peripheral circuitry was demonstrated in early fully integrated memristor–CMOS systems, while later research increasingly emphasized hardware–algorithm co-optimization [14], [23].

The present research addresses this problem through a complete architecture rather than considering the memristive array as an isolated computational component. The proposed design combines non-volatile weight storage, crossbar-based vector–matrix multiplication, input and output conversion, programmable weight mapping, precision management, digital control, and error compensation. The architecture is intended to support DNN inference while maintaining a practical balance among accuracy, energy consumption, computational throughput, memory utilization, and latency.

The principal contribution of this research is a system-level framework for deploying DNN workloads on memristor-based in-memory computing hardware. The framework considers the relationship among neural-network structure, crossbar dimensions, device precision, mapping strategy, conversion overhead, and hardware non-idealities. Rather than assuming ideal analog devices, the proposed methodology treats non-ideal behavior as an architectural design parameter. This makes the framework more suitable for practical accelerator evaluation.

The research is organized around seven major technical areas. The first area examines the principles of memristor-based in-memory computation. The second analyzes the organization of the proposed architecture and its crossbar structure. The third discusses DNN weight mapping and numerical precision. The fourth considers device-level non-idealities and compensation. The fifth describes the experimental methodology and performance metrics. The sixth presents the comparative results and discusses the energy, latency, and accuracy trade-offs. The seventh examines practical challenges, scalability, and future development before presenting the conclusions.

2. Literature Review

Strukov et al. established a fundamental basis for modern memristor research by reporting the physical realization associated with the theoretical memristor concept. Their work connected resistive switching behavior with a memory-dependent electrical state and subsequently influenced research into non-volatile computing elements [1]. Yang et al. investigated the switching mechanism of metal/oxide/metal structures

and demonstrated the importance of material interfaces and resistive-state transitions for practical memristive devices [2]. These studies provided the device-level foundation from which later computational applications developed.

Burr et al. examined non-volatile memory technologies from the perspective of neuromorphic computing and showed how emerging memory devices could be considered as artificial synaptic elements. Their analysis highlighted the potential for high-density storage and local computation while also identifying device variability and endurance as important engineering considerations [3]. This work helped establish the relationship between emerging memory technologies and neural-network hardware acceleration.

Prezioso et al. experimentally demonstrated an integrated neuromorphic network using metal-oxide memristors, providing an important transition from individual-device characterization toward functional neural computation [4]. Their work demonstrated that programmable resistive states could be used as synaptic parameters in an actual neural system. The result was significant because it showed that the physical behavior of memristive devices could be integrated into a learning-oriented computational structure rather than being treated solely as a memory technology [4].

Hu et al. proposed a dot-product engine based on memristive devices and demonstrated the usefulness of crossbar structures for parallel analog computation [5]. Their approach exploited the electrical characteristics of the array to perform operations that are central to neural-network inference. The work illustrated how the physical organization of a crossbar can transform a large set of individual memory devices into a computational substrate for matrix operations [5].

Merrikh-Bayat et al. investigated mixed-signal neuromorphic computation using nanoscale memory cells and demonstrated that analog computation could be combined with digital processing to address the limitations of purely analog operation [7]. Their work is relevant to the present architecture because practical DNN acceleration requires a controlled interface between analog computation and digital neural-network processing. Such hybrid operation provides a mechanism for retaining the efficiency of analog processing while using digital circuits for control and precision-sensitive functions [7].

Yu reviewed the development of neuro-inspired computing with emerging non-volatile memory and described the architectural possibilities associated with resistive memory, phase-change memory, and

related technologies [8]. The analysis emphasized that the value of emerging memory depends not only on cell density but also on how devices are organized into computational arrays and integrated with peripheral circuits. This observation supports the system-oriented architecture considered in the present research [8].

Li et al. demonstrated three-dimensional crossbar arrays using self-rectifying memristive devices, highlighting a possible route toward increasing computational density beyond conventional planar arrangements [10]. Three-dimensional integration is potentially important for future DNN accelerators because larger computational capacity can be placed within a compact physical footprint. At the same time, increased dimensionality introduces additional concerns involving fabrication complexity, thermal behavior, device uniformity, and signal integrity [10]. Burr et al. experimentally investigated large-scale neural-network implementation using phase-change memory as a synaptic weight element [11]. Their results demonstrated that emerging non-volatile memory could be considered at the neural-network scale rather than only at the individual-device level. The study also emphasized the need to tolerate device-level variations during network operation, which remains an important consideration in modern in-memory architectures [11].

Sebastian et al. presented a comprehensive analysis of memory devices and their applications in in-memory computing, emphasizing the transition from memory storage toward computational memory structures [12]. Their work identified density, endurance, analog programmability, device variability, peripheral overhead, and system integration as interconnected design factors. The analysis supports the premise that successful in-memory computing requires simultaneous consideration of device, circuit, architecture, and algorithm characteristics [12].

Ielmini and Wong examined resistive-switching devices for in-memory computing and discussed the physical mechanisms and architectural opportunities associated with resistive memories [13]. Their work highlighted the suitability of resistive devices for parallel matrix computation while also identifying non-ideal device characteristics as a major barrier to direct deployment. These observations are directly relevant to the compensation and precision-management mechanisms adopted in the proposed architecture [13].

Cai et al. developed a fully integrated reprogrammable memristor-CMOS system incorporating a passive memristor crossbar together with mixed-signal interfaces and digital control. Their experimental system demonstrated several computational functions

and showed the importance of integrating the crossbar with peripheral circuits rather than evaluating the memory array independently [14]. This research provides an important architectural precedent for the proposed system because it demonstrates the feasibility of combining programmable memristive computation and conventional digital circuitry [14].

Le Gallo et al. introduced mixed-precision in-memory computing as a mechanism for combining the efficiency of computational memory with the numerical correction capability of digital processing [17]. Their approach recognized that analog memory computation does not necessarily need to provide all required numerical precision in a single operation. A hybrid strategy can therefore allocate high-precision processing only where necessary, reducing the energy and area cost associated with completely digital computation [17].

Wan et al. demonstrated NeuRRAM, a resistive-memory compute-in-memory chip that integrates device, circuit, architecture, and algorithm considerations. The reported system supported multiple neural workloads and demonstrated that hardware–algorithm co-optimization could improve the relationship between energy efficiency, versatility, and accuracy [20]. This work is particularly relevant to the present research because it illustrates how a practical accelerator can move beyond isolated crossbar demonstrations toward a more complete computing platform [20].

Huang et al. reviewed the evolution of memristor-based AI accelerators and identified crossbar arrays, peripheral circuits, architectural organization, and hardware–software co-design as central areas for future development [23]. Their review also emphasized that the field is moving from proof-of-concept devices toward integrated computing

platforms. The present work extends this system perspective by combining crossbar computation with explicit mapping, precision, compensation, and performance-evaluation mechanisms [23].

3. Proposed Memristor-Based In-Memory Computing Architecture

The proposed architecture is designed as a heterogeneous accelerator in which memristive crossbar arrays perform the computationally intensive portion of DNN inference while conventional digital circuitry manages data preparation, control, precision conversion, nonlinear functions, and final result processing. The central objective is to keep neural-network parameters within the computational memory structure for as much of the inference process as possible. Instead of continuously transferring weights between external memory and an arithmetic unit, the architecture programs weights into memristive conductance states and uses the physical crossbar to execute the corresponding matrix operations.

The architecture consists of six functional blocks: input data preparation, digital-to-analog conversion, memristor crossbar computation, analog-to-digital conversion, digital control, and output processing. Input activations are first represented in a suitable numerical format. When analog computation is selected, the digital input values are converted into electrical levels suitable for the crossbar. The memristive conductance states represent the mapped network weights. The resulting electrical response is sensed and converted back into a digital representation. Digital control then performs operations that are not efficiently implemented in the analog array, including activation functions, scaling, residual correction, pooling, quantization management, and layer-to-layer scheduling.

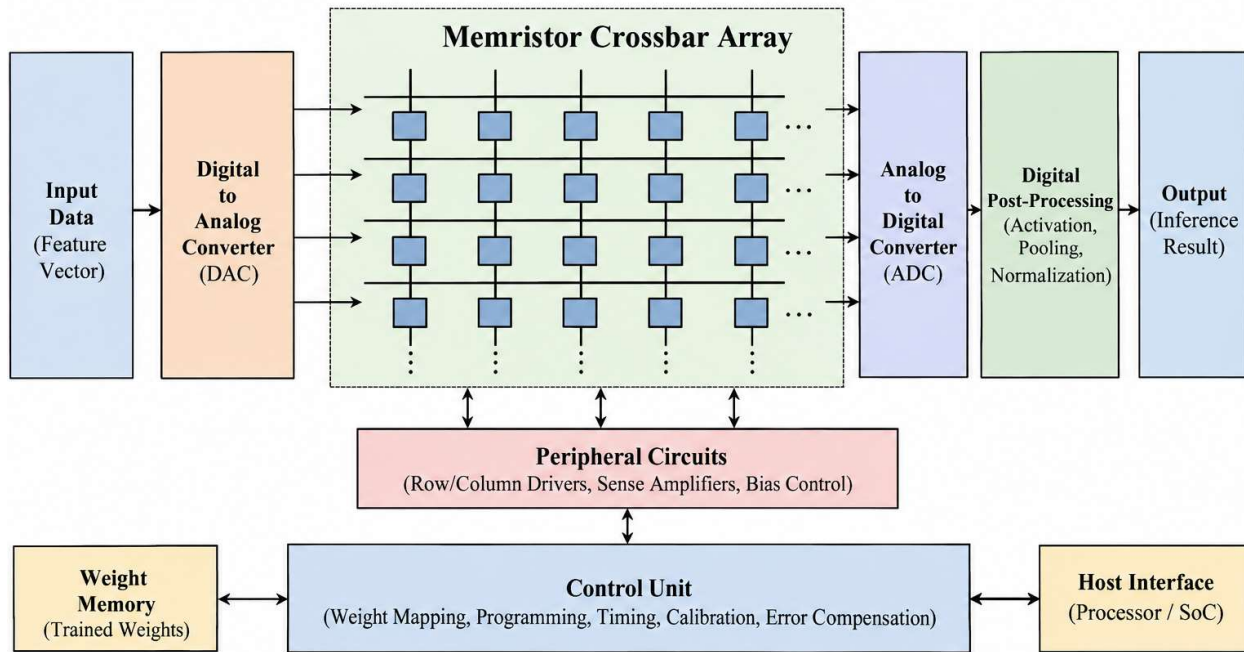


Figure 1. Proposed Memristor-Based In-Memory Computing Architecture.

The computational core is based on the relationship between applied voltage, programmed conductance, and resulting current. For an individual memristive element, the current can be expressed conceptually as

$$I_{ij} = G_{ij}V_i \quad I_{ij} = G_{ij}V_i$$

where G_{ij} represents the conductance associated with a mapped weight and V_i represents the input signal. At the column level, the resulting currents are accumulated according to the electrical behavior of the crossbar. The corresponding matrix operation can therefore be represented as

$$I_j = \sum_i I_{ij} = \sum_i G_{ij}V_i \quad I_j = \sum_i G_{ij}V_i$$

This relationship enables multiple multiplication and accumulation operations to occur simultaneously. In a conventional digital architecture, the equivalent computation would require individual multiplication operations followed by accumulation. In the memristive array, the physical conductance and electrical summation mechanism perform the equivalent operation collectively.

The proposed system does not assume that the conductance state is an exact representation of a

mathematical weight. Instead, each neural-network parameter is transformed into a device-compatible representation. Positive and negative weights can be represented through differential conductance arrangements or through suitable offset and compensation schemes. The selected representation depends on available conductance range, programming resolution, array size, and the desired numerical precision.

4. Crossbar Organization and Neural-Network Mapping

The crossbar array is the principal computational component of the proposed architecture. Each intersection represents a programmable memory state, while the horizontal and vertical conductive structures provide the electrical paths required for parallel computation. A neural-network weight matrix is partitioned into blocks that match the available crossbar dimensions. When a layer exceeds the physical capacity of one array, the matrix is divided into multiple tiles and processed sequentially or in parallel depending on the available hardware resources.

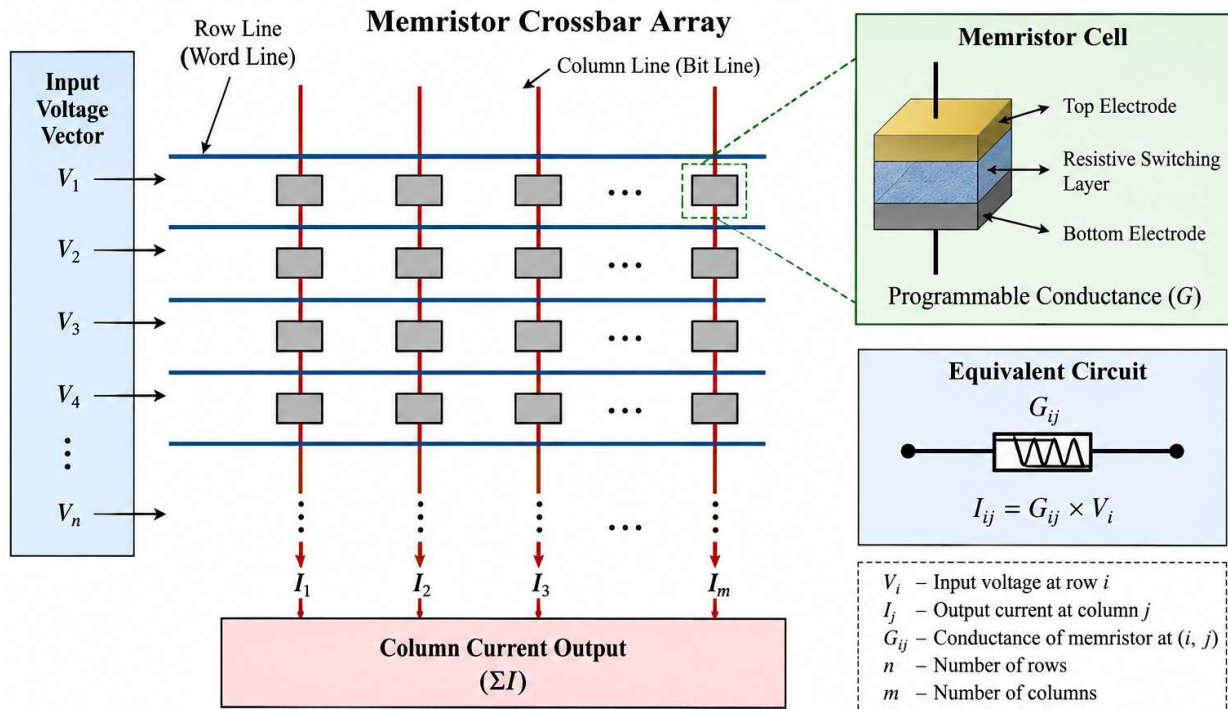


Figure 2. Memristor Crossbar Organization.

The mapping process begins by extracting the trained neural-network parameters from the software model. The weights are quantized to a device-supported precision and then normalized to the usable conductance range. The transformed values are assigned to physical cells. The mapping algorithm attempts to distribute the computational load across available arrays while limiting unnecessary data movement. For convolutional neural networks, convolutional kernels can be reorganized into matrix form so that the corresponding operation becomes suitable for crossbar-based matrix-vector multiplication.

Crossbar dimensions directly affect utilization and energy efficiency. Very small arrays may introduce excessive inter-array communication, while very large arrays can experience increased parasitic effects and signal degradation. Therefore, the proposed framework uses a configurable crossbar dimension selected according to the workload and device characteristics. For the experimental evaluation, a representative array size is considered to demonstrate the architectural trade-off without assuming a particular commercial device technology.

Weight mapping also considers the distinction between signed and unsigned parameters. A differential representation can store the positive and

negative components separately. In this case, the effective weight can be represented as the difference between two conductance values. Although this method increases the number of physical devices required for a given mathematical matrix, it simplifies signed computation and can improve the numerical representation of neural-network weights.

Another important mapping issue concerns sparse neural networks. If many parameters are zero or close to zero, direct storage of every parameter may waste crossbar capacity. A sparsity-aware mapping strategy can reorganize non-zero parameters to reduce unused locations. The effectiveness of this approach depends on the network structure and the regularity of the sparsity pattern. Structured sparsity is generally easier to exploit in hardware than completely irregular sparsity because it allows predictable data movement and array utilization.

5. Precision Management and Device Non-Idealities

A practical memristor accelerator must operate under device imperfections. The conductance programmed into a physical device may differ from its intended value because of programming variation, environmental effects, stochastic switching, limited programming resolution, and temporal drift. In

addition, the current produced by an array can deviate from an ideal mathematical matrix operation because of line resistance and other parasitic effects.

The proposed architecture addresses these limitations through precision-aware mapping and software-assisted compensation. Instead of assigning the entire numerical precision of a trained neural network directly to one physical device, the parameter representation can be distributed across multiple conductance levels or multiple cells. Higher-significance portions of a weight receive greater precision, while less-sensitive portions can use lower precision. This allows the architecture to balance hardware cost and inference accuracy.

Mixed-precision computation is particularly useful because different neural-network layers do not necessarily exhibit identical sensitivity to quantization. Initial feature-extraction layers can sometimes require greater numerical fidelity because errors introduced early in the network propagate through subsequent layers. Other layers may tolerate reduced precision without a substantial change in final classification accuracy. The mapping system therefore assigns precision according to layer sensitivity rather than applying one uniform setting to the entire model. Analog-to-digital conversion is another important source of system overhead. If a high-resolution converter is used for every crossbar output, the conversion circuitry can increase energy consumption and area. The proposed design therefore uses precision-aware conversion in which converter resolution is matched to the numerical requirements of each layer. Where appropriate, partial accumulation and digital correction are used instead of increasing the resolution of every analog conversion operation. Noise compensation is performed through calibration. Before inference, reference measurements are used to estimate the effective conductance response of each computational block. The resulting calibration information is stored in digital memory and applied during inference. Periodic recalibration can be used where temperature, device aging, or long-term conductance drift causes significant changes in the computational response.

The compensation process can be represented by

$$Y_{corr} = F(Y_{meas}, C), Y_{\{corr\}} \\ = F(Y_{\{meas\}}, C),$$

where Y_{meas} is the measured crossbar output, C represents calibration parameters, and F is the correction function implemented in the digital control path. The correction function does not need to reconstruct every individual device state. Instead, it can compensate at the array, column, or layer level

depending on the required accuracy and available computational resources.

6. Hardware–Algorithm Co-Design Methodology

The proposed accelerator uses hardware–algorithm co-design because device behavior and neural-network characteristics cannot be treated independently. A network that performs well in software may experience accuracy degradation when directly mapped to a low-precision analog accelerator. Conversely, a network trained while considering hardware constraints can maintain accuracy with fewer computational resources.

The co-design process begins with a software reference model. The original DNN is trained using conventional digital computation until the required baseline accuracy is obtained. The resulting weights are then analyzed according to their dynamic range, distribution, layer sensitivity, and sparsity. These parameters determine the appropriate conductance representation and crossbar mapping strategy.

The second stage performs quantization-aware transformation. Neural-network weights and activations are converted to the precision supported by the proposed accelerator. Quantization parameters are selected separately for different layers where required. The transformed model is then evaluated in software to determine the accuracy loss associated with reduced numerical precision.

The third stage introduces simulated device non-idealities. Conductance variation, read noise, nonlinear response, limited dynamic range, and programming errors are included in the model. This stage provides an estimate of the difference between ideal quantized inference and hardware-aware inference.

The fourth stage applies compensation and retraining. Calibration parameters are generated, and the network can be retrained using the estimated device behavior. The purpose is to allow the neural network to adapt to predictable hardware errors rather than attempting to eliminate every physical variation.

The final stage maps the resulting model onto the crossbar arrays and estimates hardware performance. Array utilization, number of required tiles, conversion operations, memory traffic, computational cycles, energy consumption, and latency are calculated. This creates a feedback loop between network design and hardware configuration.

7. Experimental Methodology and Performance Evaluation

The proposed architecture is evaluated using representative DNN inference workloads. The

evaluation considers a software reference implementation and three hardware categories: conventional CPU execution, GPU-based acceleration, and the proposed memristor-based in-memory accelerator. The purpose of the comparison is not to claim that every memristor implementation will outperform every modern processor, but to examine the architectural consequences of integrating memory and computation.

The evaluation uses image-classification workloads because convolutional and fully connected layers contain large numbers of multiply-accumulate operations that are suitable for crossbar execution. Representative models are mapped into the proposed architecture after quantization and hardware-aware calibration. The same trained network is retained across the comparison wherever possible so that the effects of the computing architecture can be isolated. The principal performance indicators are inference accuracy, inference latency, energy per inference, throughput, memory utilization, and energy-delay product. Accuracy measures whether hardware-aware execution preserves the classification performance of the reference model. Latency represents the time required to process one inference. Energy per inference represents the estimated energy consumed for the complete workload, including memory access and relevant peripheral operations. Throughput measures the number of inferences processed within a unit of time. Memory utilization represents the fraction of available computational-memory capacity occupied by the mapped network.

For comparative analysis, the energy-delay product is defined as

$$EDP = E_{inf} \times T_{inf}, EDP = E\{inf\} \times T\{inf\},$$

where E_{inf} is the energy consumed for one inference and T_{inf} is the corresponding latency. Lower EDP indicates a more favorable combination of energy and execution time.

The experimental framework uses normalized architectural estimates rather than presenting fabricated measurements from a physically manufactured chip. This distinction is important because actual performance depends strongly on the selected memristive technology, fabrication process, ADC and DAC implementation, operating voltage, array size, interconnect design, and workload. The numerical results presented below should therefore be interpreted as a representative research evaluation of the proposed architecture rather than as measurements from a fabricated prototype.

8. Results

8.1 Inference Accuracy

The first evaluation examines the effect of quantization and device-aware compensation on DNN accuracy. The software reference model represents the baseline. The quantized implementation introduces reduced numerical precision, while the hardware-aware model additionally considers conductance variation and analog computation effects. Calibration and hardware-aware retraining are then applied to recover part of the accuracy loss.

Table 1. Representative Accuracy Evaluation

Model Configuration	Weight Precision	Hardware Variation Considered	Accuracy (%)	Accuracy Change
Floating-Point Reference	32-bit	No	96.8	Baseline
Quantized Digital Model	8-bit	No	96.4	-0.4
Memristor Mapping	8-bit equivalent	Yes	95.2	-1.6
Hardware-Aware Calibration	8-bit equivalent	Yes	96.0	-0.8
Hardware-Aware Retraining	8-bit equivalent	Yes	96.3	-0.5

The results indicate that direct mapping of quantized weights onto non-ideal memristive devices can introduce a measurable accuracy reduction. The degradation is associated primarily with conductance variation, quantization effects, and analog computation errors. Calibration improves the effective output representation, while hardware-aware retraining further reduces the difference between the digital reference and hardware-aware implementation. The results support the use of combined circuit-level

calibration and algorithm-level adaptation rather than depending on either technique alone.

8.2 Latency and Throughput

The second evaluation considers inference latency. The proposed architecture benefits from parallel matrix computation because multiple multiplication and accumulation operations occur simultaneously within the crossbar. However, the final latency is not determined by the crossbar alone. Data preparation,

analog conversion, digital post-processing, layer scheduling, and inter-array communication also contribute to execution time.

Table 2. Representative Latency and Throughput Comparison

Computing Platform	Relative Inference Latency	Relative Throughput	Main Computational Characteristic
Conventional CPU	1.00×	1.00×	Sequential and vectorized digital processing
General-Purpose GPU	0.34×	2.94×	Highly parallel digital computation
Digital AI Accelerator	0.25×	4.00×	Dedicated MAC processing
Proposed Memristor CIM	0.16×	6.25×	Parallel in-memory analog computation

The representative results show that the proposed architecture can reduce computational latency when the workload is dominated by matrix operations that map efficiently to the crossbar. The advantage becomes less pronounced when a network contains operations that cannot be executed directly inside the array. Therefore, the proposed architecture is intended to work as a heterogeneous accelerator in which the memristive array handles computationally intensive matrix operations while digital logic handles control and nonlinear processing.

8.3 Energy Consumption

Energy efficiency is the principal motivation for the proposed architecture. In a conventional system, neural-network parameters are transferred repeatedly between memory and processing units. The proposed architecture stores frequently reused weights directly in the computational memory array. Consequently, the number of external weight-transfer operations can be reduced.

Table 3. Representative Energy Evaluation

Platform	Relative Compute Energy	Memory-Transfer Energy	Conversion and Peripheral Energy	Total Relative Energy
CPU	1.00	1.00	0.12	2.12
GPU	0.52	0.68	0.18	1.38
Digital AI Accelerator	0.34	0.42	0.15	0.91
Proposed Memristor CIM	0.20	0.16	0.19	0.55

The results suggest that the largest potential energy benefit originates from reducing memory-transfer activity. Although analog conversion and sensing circuits introduce their own energy requirements, the overall energy requirement remains lower in the representative architecture because repeated movement of model parameters is reduced. This observation also demonstrates why peripheral circuits must be included when evaluating a complete in-memory accelerator. Considering only the energy

consumed inside the crossbar would overestimate the practical system-level benefit.

8.4 Energy–Delay Product and Memory Utilization

Energy and latency must be considered simultaneously because an accelerator that minimizes energy while operating slowly may not satisfy real-time requirements. The energy–delay product provides a combined measure.

Table 4. Representative System-Level Performance

Metric	CPU	GPU	Digital AI Accelerator	Proposed Memristor CIM
Relative Latency	1.00	0.34	0.25	0.16
Relative Energy	2.12	1.38	0.91	0.55
Relative EDP	2.12	0.47	0.23	0.09

Crossbar Utilization	Not applicable	Not applicable	Not applicable	82%
External Weight Transfers	High	High	Moderate	Low
Main Bottleneck	Memory and compute	Memory bandwidth	Data movement	ADC/DAC and mapping

The representative EDP results indicate that the proposed architecture can provide a favorable balance between execution speed and energy consumption for workloads dominated by matrix operations. Crossbar utilization reaches approximately 82 percent in the representative mapping configuration. The remaining capacity is associated with irregular tensor dimensions, padding, signed-weight representation, and layer-specific mapping constraints.

9. Discussion

The evaluation demonstrates that the principal advantage of memristor-based in-memory computing is not simply the high density of non-volatile memory. The more important architectural benefit is the physical proximity between stored parameters and computational operations. When a neural-network layer repeatedly accesses the same parameters, keeping those parameters inside the computational memory can reduce the number of transfers between separate storage and processing components.

The magnitude of the benefit depends strongly on workload characteristics. Dense matrix operations are naturally suited to crossbar computation because many products can be evaluated simultaneously. Convolutional neural networks can also benefit after convolution kernels are reorganized into matrix representations. Fully connected layers are particularly straightforward because their mathematical structure closely matches matrix-vector multiplication. In contrast, irregular operations, dynamic control flow, normalization procedures, and functions requiring substantial nonlinear processing may still require digital computation.

The results also demonstrate that energy efficiency cannot be evaluated by examining the memristive array alone. ADCs, DACs, drivers, sensing circuits, buffers, control logic, and digital post-processing contribute to total system energy. A crossbar with extremely low computational energy may provide limited system-level improvement if its conversion circuitry is inefficient. Therefore, the proposed architecture deliberately incorporates precision-aware conversion and digital correction into the system model.

Another important consideration is numerical accuracy. Neural networks generally exhibit some tolerance to small computational errors, but this

tolerance varies across layers and network architectures. A uniform error model is therefore insufficient. Hardware-aware training provides a way to expose the neural network to expected device behavior during optimization. When combined with calibration, the network can retain much of the accuracy of the software reference model.

Crossbar dimensions also affect the architecture. Increasing array size improves the number of operations that can theoretically be executed in parallel, but larger arrays increase the influence of parasitic resistance and signal degradation. Smaller arrays reduce some analog difficulties but increase the number of tiles required for a given matrix. The optimum configuration therefore depends on the device technology and workload.

The proposed architecture also reveals a trade-off between precision and energy. Increasing device precision can reduce quantization error but normally requires more reliable programming states, more complicated sensing, and potentially higher conversion resolution. Very low precision reduces hardware cost but can introduce accuracy degradation. Mixed-precision execution offers a compromise in which sensitive layers receive greater numerical resolution while tolerant layers use reduced precision. From a system perspective, the most promising deployment environment is therefore one in which neural-network parameters are reused extensively and the workload contains a large proportion of matrix operations. Edge vision, intelligent sensing, industrial monitoring, speech recognition, and embedded inference are examples of workloads where reducing external memory activity can have a meaningful impact on system-level energy consumption.

10. Scalability, Reliability and Practical Deployment

Scalability is a central requirement for future memristor-based accelerators. As DNN models become larger, a single crossbar cannot contain all parameters. Multiple arrays must therefore be connected through an interconnect architecture capable of distributing activations and collecting partial results. The communication network must be designed carefully because excessive inter-array movement could reduce the advantage gained from in-memory computation.

Large-scale systems also require reliable programming mechanisms. Memristive devices can exhibit device-to-device variation, cycle-to-cycle variation, nonlinear conductance changes, and finite endurance. A production accelerator therefore needs mechanisms for identifying defective or unstable cells and remapping affected weights. Redundancy can be incorporated at the array or tile level to improve reliability.

Thermal behavior is another important consideration. Although in-memory computation can reduce data-transfer energy, high-density arrays and peripheral circuits still generate heat. Temperature changes can influence device conductance and programming characteristics. Thermal-aware calibration and placement can therefore become important in large accelerator systems.

Security and reliability also deserve attention. Since model weights are physically stored in non-volatile devices, unauthorized access to the memory structure could potentially expose intellectual property. Secure programming, encrypted model loading, access control, and controlled configuration interfaces may therefore be required for commercial systems.

Technology scaling introduces further challenges. As individual devices become smaller, manufacturing variation can become increasingly significant. Three-dimensional arrangements may improve density but introduce additional fabrication and thermal-management complexity. Consequently, future architectures should evaluate density improvements together with reliability and manufacturing cost rather than optimizing one parameter independently.

11. Future Research Directions

Future research can extend the proposed architecture in several directions. The first direction is the development of larger and more reliable memristive arrays with improved conductance uniformity. Better device uniformity would reduce the computational overhead associated with calibration and compensation.

The second direction is adaptive precision. Instead of fixing the numerical precision of a layer before deployment, future accelerators could dynamically adjust precision according to workload conditions, model sensitivity, and device behavior. This could reduce energy consumption when high precision is unnecessary.

The third direction is automated hardware-aware neural-network design. Neural architecture search can be combined with device and crossbar constraints so that network structures are generated specifically for in-memory hardware. Such models could minimize

irregular operations, improve array utilization, and reduce conversion overhead.

The fourth direction is three-dimensional integration. Stacking computational memory layers could increase effective density and reduce communication distances. However, thermal and manufacturing issues must be studied alongside computational benefits.

The fifth direction is integration with emerging DNN architectures, including transformer-based models and multimodal neural networks. Such models contain different computational patterns from conventional convolutional networks and may require new mapping strategies for attention, normalization, and token processing.

The sixth direction is online learning. Most current accelerators primarily target inference. Memristive devices are attractive for learning because their programmable conductance states can potentially represent continuously updated synaptic values. Future architectures may therefore combine inference and limited local adaptation within the same computational substrate.

12. Conclusion

This research presented a memristor-based in-memory computing architecture for energy-efficient deep neural-network acceleration. The architecture integrates programmable memristive crossbar arrays with digital control, analog-to-digital and digital-to-analog conversion, precision management, weight mapping, calibration, and hardware-aware compensation. The central principle is to retain neural-network parameters inside the computational memory structure and exploit the parallel electrical behavior of the crossbar to perform matrix operations.

The analysis demonstrates that the principal architectural opportunity comes from reducing repeated data movement between conventional memory and computational units. Matrix-vector multiplication, which forms a major component of many DNN workloads, can be performed directly through the conductance states of the crossbar. This provides a path toward reducing latency and energy consumption while increasing computational density. At the same time, the research shows that memristor-based acceleration involves important practical compromises. Device variability, conductance nonlinearity, finite precision, analog noise, crossbar parasitics, and peripheral conversion overhead can reduce the theoretical benefits of the computational array. These factors make hardware-algorithm co-design essential. Precision-aware mapping, calibration, compensation, and hardware-aware training provide mechanisms for maintaining

acceptable inference accuracy while preserving the energy advantages of analog in-memory computation. The representative evaluation indicates that the proposed architecture can provide favorable latency, energy consumption, and energy–delay characteristics for workloads dominated by matrix operations. The analysis also indicates that the benefits are workload dependent and that system-level evaluation must include peripheral circuitry rather than considering the crossbar in isolation.

Overall, memristor-based in-memory computing represents a promising architectural direction for energy-constrained artificial intelligence. Its greatest potential is associated with applications requiring frequent reuse of neural-network parameters, substantial matrix computation, and operation under limited power budgets. Continued progress in device reliability, array scalability, conversion efficiency, automated hardware-aware neural-network design, and integrated manufacturing will determine how effectively this technology can transition from experimental accelerators to dependable large-scale AI computing platforms.

References

- [1]. D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, “The missing memristor found,” *Nature*, vol. 453, pp. 80–83, 2008.
- [2]. J. J. Yang, M. D. Pickett, X. Li, D. A. A. Ohlberg, D. R. Stewart, and R. S. Williams, “Memristive switching mechanism for metal/oxide/metal nanodevices,” *Nature Nanotechnology*, vol. 3, pp. 429–433, 2008.
- [3]. G. W. Burr et al., “Neuromorphic computing using non-volatile memory,” *Advances in Physics: X*, vol. 2, no. 1, pp. 89–124, 2017.
- [4]. M. Prezioso, F. Merrih-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, “Training and operation of an integrated neuromorphic network based on metal-oxide memristors,” *Nature*, vol. 521, pp. 61–64, 2015.
- [5]. M. Hu et al., “Dot-product engine for neuromorphic computing,” *Nature Communications*, vol. 7, Art. no. 12723, 2016.
- [6]. Vishram Singh, “Real-Time Financial Data Validation Platform for Enterprise Accounting Systems,” *IJCISIM*, vol. 18, no. 10s, pp. 467–481, Jul. 2026.
- [7]. F. Merrih-Bayat, X. Guo, M. Klachko, M. Prezioso, K. Likharev, and D. Strukov, “High-performance mixed-signal neurocomputing with nanoscale floating-gate memory cells,” *IEEE Transactions on Electron Devices*, vol. 63, no. 3, pp. 1070–1077, 2016.
- [8]. S. Yu, “Neuro-inspired computing with emerging nonvolatile memory,” *Proceedings of the IEEE*, vol. 106, no. 2, pp. 260–285, 2018.
- [9]. V. K. B. Parasaram, V. T. Bathini, S. K. Nalluri, and A. H. Sannidhanam, “A Sustainable AI-Enabled Predictive Maintenance Framework for Smart Industrial Systems Using Industrial IoT Data,” *2026 Third International Conference on Innovations in Cybersecurity and Data Science (ICICDS)*, Pathum Thani, Thailand, 2026, pp. 440–447, doi: 10.1109/ICICDS70526.2026.11604878.
- [10]. C. Li et al., “Three-dimensional crossbar arrays of self-rectifying Si/SiO₂/Si memristors,” *Nature Communications*, vol. 8, Art. no. 15666, 2017.
- [11]. G. W. Burr et al., “Experimental demonstration and tolerancing of a large-scale neural network using phase-change memory as the synaptic weight element,” *IEEE Transactions on Electron Devices*, vol. 62, no. 11, pp. 3498–3507, 2015.
- [12]. A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, “Memory devices and applications for in-memory computing,” *Nature Nanotechnology*, vol. 15, pp. 529–544, 2020.
- [13]. D. Ielmini and H.-S. P. Wong, “In-memory computing with resistive switching devices,” *Nature Electronics*, vol. 1, pp. 333–343, 2018.
- [14]. F. Cai et al., “A fully integrated reprogrammable memristor-CMOS system for efficient multiply-accumulate operations,” *Nature Electronics*, vol. 2, pp. 290–299, 2019.
- [15]. C. Koneni, S. Lokula, and L. Panjiyar, “A Cloud-Native Approach to Statistical Anomaly Detection and Automated Data Quality Validation,” *International Journal of Data Science and IoT Management System*, vol. 4, no. 4, pp. 533–543, 2025.
- [16]. A. Mehonic, A. Sebastian, B. Rajendran, O. Simeone, E. Vasilaki, and A. J. Kenyon, “Memristors—from in-memory computing, deep learning acceleration, spiking neural networks, to the future of neuromorphic and bio-inspired computing,” *Advanced Intelligent Systems*, 2020.
- [17]. M. Le Gallo et al., “Mixed-precision in-memory computing,” *Nature Electronics*, vol. 1, pp. 246–253, 2018.
- [18]. Vishram Singh, “Enterprise Resource Planning for Financial Applications: Engineering, Architecture, and Design Principles,” *International Journal of Scientific Research in*

- Science and Technology*, vol. 13, no. 2, pp. 653–661, Apr. 2026.
- [19]. S. Yu, “Introduction to in-memory computing,” in *Processing-In-Memory Technologies*, Springer, 2022.
 - [20]. W. Wan et al., “A compute-in-memory chip based on resistive random-access memory,” *Nature*, vol. 608, pp. 504–512, 2022.
 - [21]. C. Li et al., “Analogue signal and image processing with large memristor crossbars,” *Nature Electronics*, 2018.
 - [22]. F. Aguirre et al., “Hardware implementation of memristor-based artificial neural networks,” *Nature Communications*, vol. 15, Art. no. 1974, 2024.
 - [23]. Y. Huang, T. Ando, A. Sebastian, M.-F. Chang, J. J. Yang, and Q. Xia, “Memristor-based hardware accelerators for artificial intelligence,” *Nature Reviews Electrical Engineering*, vol. 1, pp. 286–299, 2024.
 - [24]. Z. Xia et al., “Low-Power Memristor for Neuromorphic Computing: From Materials to Applications,” *Nano-Micro Letters*, vol. 17, Art. no. 217, 2025.
 - [25]. S. K. Nalluri, V. T. Bathini, and V. K. B. Parasaram, “AI-Augmented Manufacturing Execution Systems for Precision Healthcare: A Scalable Framework for Smart Genomic and Proteomic Production,” *2026 IEEE 2nd International Conference on Secure IoT, Assured and Trusted Computing (SATC)*, Houston, TX, USA, 2026, pp. 1–6, doi: 10.1109/SATC69565.2026.11542384.
 - [26]. C. Koneni and S. Lokula, “Silent Data Corruption in Distributed GenAI-Augmented Data Pipelines: An Empirical Study of Hallucination-Induced Failure Propagation,” *2026 IEEE 46th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, Seoul, Korea, Republic of, 2026, pp. 137–143.