Check for updates

Advanced Electronic Materials

ADVANCED ELECTRONIC MATERIALS Open Access

www.advelectronicmat.de

RESEARCH ARTICLE OPEN ACCESS

Predictable and Scalable Analog Matrix–Vector Multiplication in Memristor Crossbars via Closed-Form Wire-Resistance Compensation

Davide Rossetti[1,2] | Yichun Xu[2] | Ludovica Asselta[1] | J. Joshua Yang[2] | R. Stanley Williams[2] | Fernando Corinto[1]

[1]Department of Electronics and Telecommunications Engineering, Politecnico di Torino, Turin, Italy | [2]Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, California, USA

Correspondence: Davide Rossetti (davide.rossetti@polito.it)

Received: 17 March 2026 | **Revised:** 13 August 2026 | **Accepted:** 29 August 2026

Keywords: edge AI | neuromorphic computing | computation complexity | mathematical modeling | crossbar array | wire resistance | matrix vector multiplication | memristor | classification

ABSTRACT

Wire resistance in memristor crossbars introduces geometry-dependent voltage attenuation that degrades analog matrix–vector multiplication (MVM), while parasitic sneak paths further distort output currents as array dimensions scale. Here, we develop a closed-form distributed line-resistance model that jointly accounts for horizontal and vertical interconnects as well as parasitic conduction, yielding explicit node-voltage profiles and a compact analytical description of how interconnect losses modify the effective conductance matrix. Leveraging this predictive framework, computation is kept fully analog and two low-overhead correction strategies are introduced: (1) a geometry-aware adjustment of the programmed conductances derived directly from the analytically computed wire-resistance error; and (2) a four-configuration symmetry-averaging scheme that suppresses residual spatial non-uniformities without requiring any modification to peripheral circuits. Circuit- and task-level evaluations on crossbar arrays demonstrate that, once compensated, MVM errors remain bounded and no longer scale with array size, while accuracy is preserved with a constant corrective overhead of four MVMs per elementary array block, validated up to 128×128 arrays. By fixing the problem class, correction targets are computed analytically and reused, replacing iterative tuning with predictable operation and enabling reliable analog in-memory computing at scale.

1 | Introduction

Memristor crossbars are increasingly recognized as promising candidates for accelerating computations in linear algebra, as they offer significant advantages in terms of parallel computing capability and data storage [1–3]. The array-based structure of these devices, illustrated in Figure 1, enables massive parallel processing and reduces the power overhead associated with data migration, which remains a significant bottleneck in conventional computing architectures [4, 5]. This makes memristor crossbars particularly attractive for neuromorphic and edge-computing deployments [4, 6, 7], where efficiency and speed are paramount, especially since matrix–vector multiplication (MVM)—a fundamental linear-algebra operation used in machine learning, signal processing, and scientific computing—scales as $\mathcal{O}(n^2)$ in digital systems, whereas the crossbar's inherent parallelism reduces its effective latency to a single physical step [8–12]. Despite their potential, the performance of memristor crossbars is significantly compromised by the inherent resistance of the metal wires connecting the memristor cells. As the size of

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.
© 2026 The Author(s). *Advanced Electronic Materials* published by Wiley-VCH GmbH

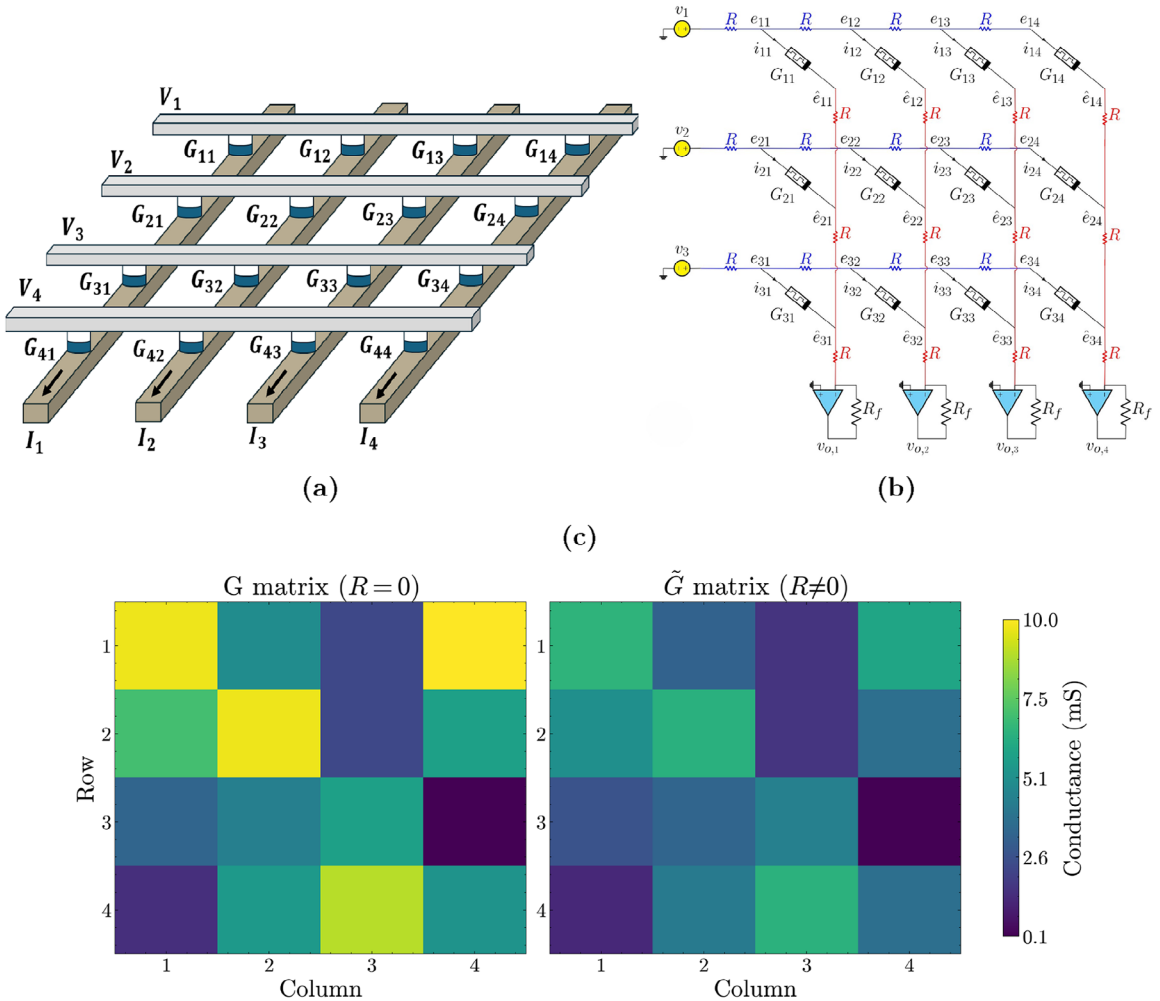

FIGURE 1 | Crossbar operation, circuit implementation, and effect of wire resistance. (a) A 4×4 memristive crossbar highlighting the technical operating principle. (b) Circuit implementation: schematic of an $m \times n$ memristive crossbar ($m \neq n$) with wire resistance R to emphasize generality beyond square arrays. (c) Impact of wire resistance: the programmed conductances in the non-ideal case (second matrix, $R \neq 0$) are degraded relative to the ideal case (first matrix, $R = 0$). Results shown were obtained with the following simulation parameters: $G_{\min} = 0.1$ mS, $G_{\max} = 10$ mS, $R = 5\ \Omega$, $n = 4$.

the crossbar increases, the resistance along both the horizontal and vertical wires leads to significant voltage drops. These voltage drops can severely affect the precision of analog calculations and lead to inaccurate results, especially in large systems [13, 14]. In addition, parasitic conduction through non-selected devices (sneak paths) further perturbs current flow [15, 16]. Although several strategies have been proposed to mitigate this problem, including compensation algorithms and digital corrections, these approaches often fall short; digital corrections in particular require precomputation and adaptation of input signals, which can negate much of the efficiency achieved by analog processing in the first place [17–19]. This work introduces a closed-form mathematical framework that rigorously captures the impacts of horizontal and vertical line resistance, as shown in Figure 1, thereby enabling accurate matrix–vector multiplication and suppressing sneak-path currents. Beyond furnishing the effective conductance matrix, the model quantitatively characterizes wire-resistance–induced degradation. Concretely, it provides (1) the effective conductance mapping that incorporates both horizontal and vertical line resistances, (2) the node-voltage distribution along rows and columns, and (3) the presence of sneak paths. This view exposes how edge and corner regions of the crossbar suffer larger IR drops, leading to spatially non-uniform errors in the measured output currents. Leveraging this quantitative description, we derive a corrective strategy: a four-matrix geometry-averaging scheme that symmetrizes residual, position-dependent errors by combining outputs from four geometric transformations of the original matrix. Together, these steps substantially reduce MVM error while preserving in-situ analog parallelism. The mathematical approach presented improves the precision of the final output and, in turn, the reliability of analog MVMs, while remaining aligned with recent demonstrations of analog-grade large crossbars and high-precision programming protocols [20]. We substantiate the practicality of the proposed framework through a multi-level evaluation strategy. Circuit-level simulations explicitly resolve the impact of horizontal and vertical line resistance and validate the proposed correction mechanisms. At the application level, we consider both structured linear-algebra workloads and data-driven inference tasks. In particular, on a physical memristor crossbar we evaluate experimentally, on hardware, a 64×64 discrete Fourier transform (DFT) to probe error accumulation in large-scale matrix–vector multiplications,

2199160x, 0, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aelm.70580 by Cochraneltalia, Wiley Online Library on [24/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

and we assess a multi-layer perceptron performing handwritten digit classification on the MNIST dataset using the IBM Analog Hardware Acceleration Kit (AIHWKit) [21, 22], whose inference noise models explicitly account for programming noise, read noise, and temporal conductance drift calibrated on measured device statistics.

These results demonstrate that wire resistance acts as a predictable and correctable geometric transform: once analytically modeled and compensated, it enables accurate in situ linear algebra and stable neural-network inference at scale, thereby moving memristor crossbars closer to practical neuromorphic and edge-AI deployment.

2 | Results

2.1 | Wire Resistance Effects in Crossbar

In large memristive crossbars, the assumption of ideal, perfectly conducting wires breaks down, since every line of the array is segmented by a finite interconnect resistance that distorts the applied voltages and currents. These parasitics give rise to two complementary effects: vertical wire resistance, where currents traveling down the sensing lines toward the output amplifiers cause progressive voltage drops that reduce the effective readout current seen by deeper devices in each column; and horizontal wire resistance, where the input signal applied at the edge of a row is progressively attenuated as it propagates so that devices farther from the input receive a lower potential than intended. Both contributions result in systematic, position-dependent errors that grow with array dimensions, even if devices are perfectly programmed, as can be observed in Figure 1. This motivates a rigorous formulation of the MVM operation in the presence of wire resistance, which we develop by considering a memristive crossbar with a conductance matrix $\mathbf{G} \in \mathbb{R}^{n\times m}$, where for simplicity—and in line with typical experimental demonstrations—we can assume $n = m$, while the general theoretical case $n \neq m$ is fully developed in this section. As illustrated in Figure 1, we denote by $\underline{\mathbf{i}} = (\mathbf{i}_1, \ldots, \mathbf{i}_m)^T \in \mathbb{R}^{nm}$ the vector of all device currents, where each $\mathbf{i}_k$ collects the n currents through the memristors of the kth column; by $\underline{\mathbf{e}} = (\mathbf{e}_1, \ldots, \mathbf{e}_m)^T$ and $\underline{\hat{\mathbf{e}}} = (\hat{\mathbf{e}}_1, \ldots, \hat{\mathbf{e}}_m)^T$ the vectors of node voltages at the upper and lower terminals of the memristors, respectively; by R_{col} and R_{row} the uniform interconnect resistances along columns and rows; and by $\mathbf{G}_{\text{diag}} = \text{diag}(\mathbf{G}_1, \ldots, \mathbf{G}_m) \in \mathbb{R}^{nm\times nm}$ the block-diagonal matrix collecting the conductances $\mathbf{G}_k$ of each column. The applied input is described by $\mathbf{v} = (v_1, \ldots, v_n)^T \in \mathbb{R}^n$, which generates the augmented vector $\underline{\underline{\mathbf{v}}} = (\mathbf{v}, \ldots, \mathbf{v})^T \in \mathbb{R}^{nm}$. In the ideal case without wire resistance, the crossbar implements the standard matrix–vector multiplication

$$\mathbf{i}_{\text{ideal}}^{out} = \mathbf{G}^T\mathbf{v},$$

which is equivalently expressed in block form as

$$\begin{cases} \underline{\mathbf{i}} = \mathbf{G}_{\text{diag}}\underline{\underline{\mathbf{v}}}, \\ i_k^{out} = \mathbf{1}_k^T\underline{\mathbf{i}} = \sum_{j=1}^{n} i_{jk}, \end{cases}$$

where $\mathbf{1}_k^T$ selects the currents of the kth column currents. For compactness in the following analysis, we introduce two key matrices: the sum matrix $\mathbf{L}$, which accumulates current contributions along a column, and the difference matrix $\mathbf{V}$, which represents the discrete derivative along a line,

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ 1 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 1 & 1 & \ldots & 1 \end{pmatrix}, \qquad \mathbf{V} = \begin{pmatrix} -1 & 1 & \ldots & 0 \\ 0 & -1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 0 & \ldots & 0 & -1 \end{pmatrix} = -(\mathbf{L}^T)^{-1}.$$

With these definitions in place [13], to capture the effect of vertical interconnect resistance we consider the relation between the applied node voltages and the resulting device currents within each single column:

$$\underline{\mathbf{i}} = \left[\mathbf{G}_{\text{diag}}^{-1} + R_{\text{col}}(\mathbb{I}_{m\times m} \otimes \mathbf{L}^T\mathbf{L})\right]^{-1}\underline{\mathbf{e}}, \tag{1}$$

Equation (1) reveals that vertical parasitics cause a systematic attenuation of device currents that depends on their location within the column, as fully derived in the Section S1. In addition to vertical losses, complementary degradation occurs along the horizontal direction. The resulting node potentials satisfy a recursive relation that can be recast in matrix form as

$$[(\mathbf{V}\mathbf{V}^T \otimes \mathbf{I}_n) + R_{\text{row}}\mathbf{H}^{-1}]\,\underline{\mathbf{e}} = \underline{\mathbf{u}}_1, \tag{2}$$

where $\mathbf{H} = \text{diag}(\mathbf{G}_1^{-1} + R_{col}\mathbf{L}^T\mathbf{L}, \ldots, \mathbf{G}_m^{-1} + R_{col}\mathbf{L}^T\mathbf{L})$ is the block matrix that embeds the vertical wire resistance contribution derived above, and $\underline{\mathbf{u}}_1 = (\mathbf{v}, 0, \ldots, 0)^T \in \mathbb{R}^{nm}$ represents the input vector, corresponding to the application of the voltage $\mathbf{v}$ at the first column while keeping all other entries zero, i.e., the initial excitation applied to the crossbar (see Section S2). This formulation highlights that the effective voltages across devices decrease with distance from the input, producing a systematic bias along each row, as shown in Figure 2. Taken together, the vertical and horizontal contributions yield the full expression for the current distribution across the crossbar, which can be written as

$$\underline{\mathbf{i}} = [\mathbf{G}_{\text{diag}}^{-1} + \mathbf{R}_W]^{-1}\underline{\underline{\mathbf{v}}}, \tag{3}$$

with

$$\mathbf{R}_W = R_{\text{col}}(\mathbf{I}_m \otimes \mathbf{L}^T\mathbf{L}) + R_{\text{row}}(\mathbf{L}\mathbf{L}^T \otimes \mathbf{I}_n), \tag{4}$$

where $\mathbf{R}_W$ compactly accounts for both vertical and horizontal parasitic terms. Equation (3) extends the ideal MVM relation $\mathbf{i} = \mathbf{G}^T\mathbf{v}$ to the realistic case of resistive interconnects. In this formulation, the device-level conductances are explicitly coupled with the geometry-dependent voltage drops, resulting in an effective modification of the conductance matrix. The change in conductance is quantified as a percentage, as illustrated in Figure 2. The complete derivation and algebraic steps are reported in the Section S3.

2199160x, 0, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aelm.70580 by CochraneItalia, Wiley Online Library on [24/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

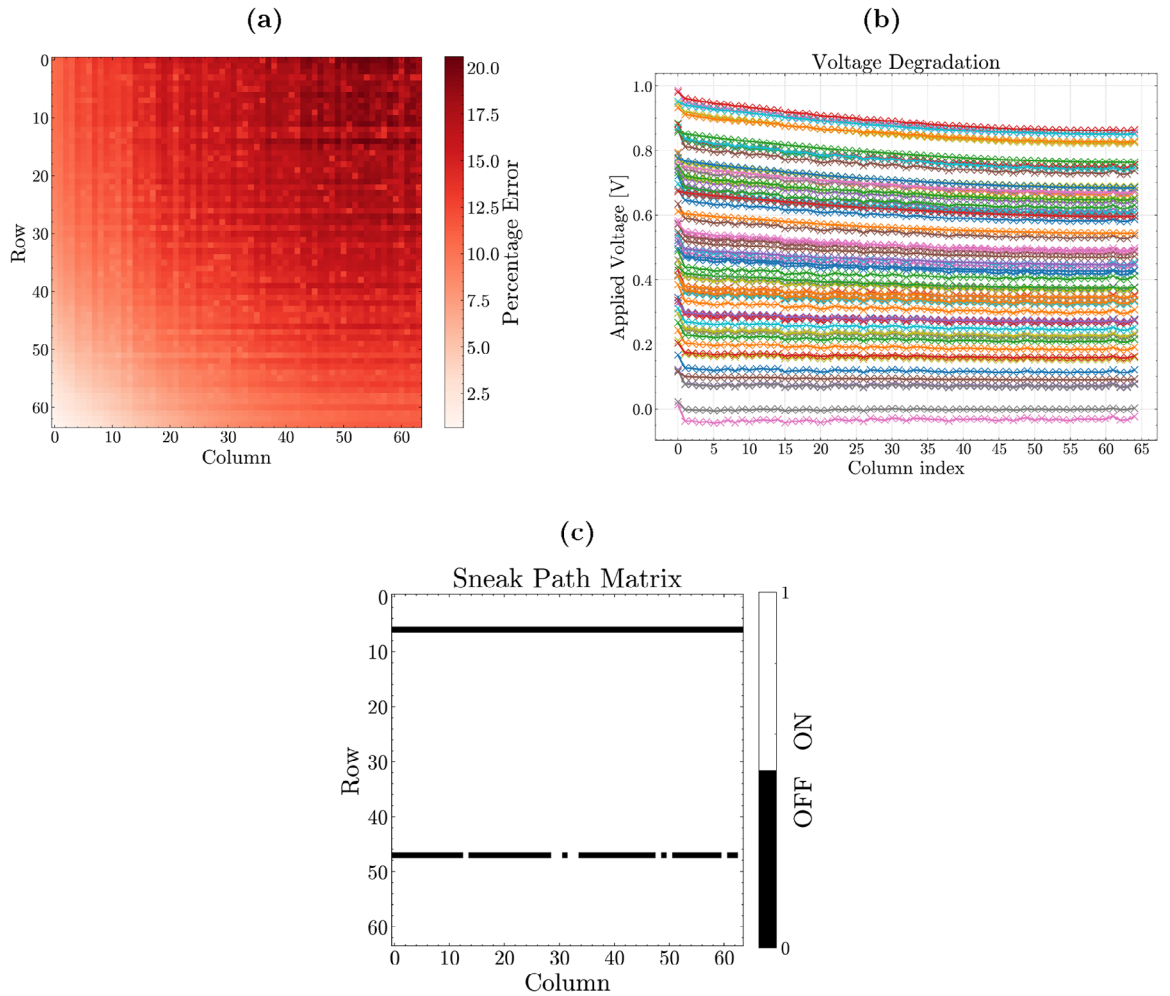

FIGURE 2 | | Error analysis, voltage degradation, and sneak path simulations in crossbar arrays. (a) Relative errors associated with the effective conductance matrix (simulation parameters: $G_{\min} = 10\ \mu S,\ G_{\max} = 100\ \mu S,\ R = 1\ \Omega,\ n = 64$). (b) Degradation of voltages along a selected row of the crossbar. (c) Simulation of sneak path currents within the array.

2.2 | Impact of Sneak Paths

Sneak paths in memristive crossbars arise when unintended current loops form between non-selected devices, producing parasitic contributions that distort readout signals and degrade computational accuracy. Their effect becomes increasingly severe as array dimensions scale, ultimately limiting both precision and energy efficiency in large systems. Based on the framework established in the previous section, the influence of sneak paths can be captured by evaluating the effective voltage distributions at the upper and lower terminals of all devices in the array. Having obtained a recursive formulation for the upper terminal voltage (Equation (2)), the corresponding lower terminal potential for each kth column can be written as

$$\hat{\boldsymbol{e}}_{\boldsymbol{k}} = (\mathbf{I}_{n\times n} + \boldsymbol{G}_{\boldsymbol{k}}^{-1}\boldsymbol{V}^{T}\boldsymbol{V})^{-1}\boldsymbol{e}_{\boldsymbol{k}}. \tag{5}$$

By comparing upper and lower terminal voltages, we can directly identify the conduction state of each device: when the voltage drop is positive, the memristor conducts as intended; when it becomes negative, the current reverses direction, thereby revealing a sneak path contribution. This analysis enables the identification of which conductances actively participate in the intended vector–matrix multiplication and which are involved in parasitic paths, thereby highlighting the systematic errors introduced by sneak currents. A representative example is shown in Figure 2, where ON and OFF states are mapped from numerical simulations. Full mathematical derivations and the algorithmic implementation are reported in the Section S4. Building on these observations, it becomes clear that the presence of sneak paths does not affect all computational tasks equally, but rather depends on the class of problems considered. In particular, once a class of problems can be described by a generic matrix A, it is possible to define a corresponding set of operating conditions that renders the crossbar robust against parasitic currents. This involves carefully selecting the input voltage range and confining the programmed memristor conductances to an optimal interval, following the analytical approach described in Equation (5). Within this regime, all instances of the problem class can be computed without interference from sneak paths: conductance values are chosen such that they remain sufficiently high to guarantee measurable output currents at the peripheral operational amplifiers, while still being low enough to prevent unnecessary power dissipation. Such trade-offs in the selection of conductance levels

2199160x, 0, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aelm.70580 by CochraneItalia, Wiley Online Library on [24/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

have been extensively discussed in the literature as central to enabling reliable and energy-efficient in-memory computing [10, 20, 23, 24].

2.3 | Correction Strategies

The impact of line resistance can be quantitatively described by expanding the error term in the expression:

$$\underline{\boldsymbol{i}} = [\boldsymbol{G}_{\boldsymbol{diag}}^{-1} + \boldsymbol{R}_W]^{-1}\underline{\boldsymbol{v}}, \tag{6}$$

which, via the Woodbury identity, becomes

$$[\boldsymbol{G}_{\boldsymbol{diag}}^{-1} + \boldsymbol{R}_W]^{-1} = \boldsymbol{G}_{\boldsymbol{diag}} - \boldsymbol{G}_{\boldsymbol{diag}}\boldsymbol{R}_W\boldsymbol{G}_{\boldsymbol{diag}} + \mathcal{O}(\boldsymbol{R}_w^2). \tag{7}$$

This expansion (see Section S5 for the complete derivation) highlights that the original conductance matrix $\mathbf{G}_{\text{diag}}$ is systematically distorted by terms proportional to $\mathbf{R}_{\mathbf{W}}$. In principle, this formulation enables an analytical pre-compensation strategy, in which the programmed conductance values are adjusted entry by entry to counteract the predicted wire-resistance-induced distortion. This approach is direct and can be advantageous for sparse or highly static matrices, where only a limited number of entries need to be corrected and the compensated values can be computed and programmed once. For dense matrices, however, the correction becomes matrix-specific and must be evaluated over a large fraction of the $n \times n$ conductance entries. As a result, extensive entry-wise pre-compensation can introduce a substantial pre-processing and programming burden, reducing the practical advantage of analog in-memory computation for dense MVM workloads [25–27] (see Section S11).

Several IR-drop mitigation strategies have been proposed for memristive and resistive-crossbar accelerators, including circuit-level compensation, parasitic-resistance-aware programming, numerical de-embedding, weight remapping, and redundancy- or slicing-based precision enhancement [18, 28–34]. These approaches typically compensate wire-resistance errors by modifying the peripheral circuitry, adjusting the programmed conductances, remapping sensitive weights, or representing each value through multiple devices or slices [18, 32–34]. It is worth noting that some of these redundancy- and averaging-based strategies target mainly stochastic, device-level non-idealities such as programming variability and stuck devices [33], rather than the deterministic, geometry-induced bias caused by wire resistance addressed here; for this reason, a comparison of resource requirements between the two classes of approach is reported for clarity in Section S9 (Table S2). The Four-Matrix Algorithm proposed here follows a different route: the averaging operation is dictated by the closed-form two-dimensional wire-resistance model, which shows that each matrix element experiences a distinct distortion determined jointly by its row and column position. Specifically, one corner of the array is systematically the least affected by wire-resistance-induced deviation while the diagonally opposite corner accumulates the largest distortion, with the error growing monotonically in between, as shown in Figure 2. The four transformations G_1–G_4 correspond to the dihedral group of order four acting independently on row and column indices, which is the minimal set of transformations that carries every corner of the array through all four relative error positions exactly once (see Section S9 and Figure S4 for the corresponding quantitative analysis). By applying complementary geometric transformations, each coefficient is evaluated in different regions of the same IR-drop landscape, and the reordered outputs redistribute the leading spatial bias over the full crossbar area. Thus, the method performs geometry-driven averaging of the deterministic wire-resistance distortion without requiring additional correction circuitry, iterative conductance recalibration, or network-specific sensitivity remapping. For the 64×64 arrays evaluated in this work, four configurations are sufficient to suppress the dominant spatial distortion; for larger arrays, the same symmetry-averaging principle can be extended by applying the four-configuration correction independently to array blocks whose size is set by the operating conductance range and wire resistance (Section S13 and Table S3), rather than to the array as a whole.

2.4 | Four-Matrix Algorithm

This method exploits symmetry and rotation transformations applied to the original conductance matrix, generating four distinct matrix–vector multiplications, as illustrated in Figure 3. Specifically, the second matrix corresponds to the mirror image of the original one, where an element initially located at position (i, j) is mapped to position $(n - i + 1, m - j + 1)$. The third matrix is obtained by inverting the order of the columns of the original conductance matrix (i.e., the first column becomes the last, the second becomes the penultimate, and so on), while the fourth is produced by inverting the columns of the mirrored matrix. Each transformed instance thus provides a complementary perspective on how wire resistance distorts the effective conductance distribution across the crossbar. When the results of the four evaluations are combined, the corresponding matrices (or, in hardware, the output currents) are summed and subsequently normalized by a constant factor $f(n, G_{\min}, G_{\max})$ that depends on the crossbar dimension and the conductance interval, as reported in Section S6 and Table S1.

This geometry-averaging strategy is complementary to the analytical pre-compensation discussed in Section 2.3. The pre-compensation approach attempts to correct the effective conductance matrix locally by modifying each programmed conductance according to the analytically predicted distortion. The Four-Matrix Algorithm instead avoids local conductance correction and uses geometric transformations to redistribute the deterministic spatial error over complementary positions of the same two-dimensional crossbar. A quantitative comparison of the two strategies under identical array conditions, reported in Section S11 (Figure S5), shows that both approaches achieve closely comparable peak conductance error (3.42% for pre-compensation vs. 3.88% for the Four-Matrix Algorithm), while differing substantially in computational overhead and recalibration cost: pre-compensation requires an $\mathcal{O}(n^6)$ dense computation to be fully repeated whenever the programmed matrix changes, whereas the Four-Matrix Algorithm requires only fixed index

2199160x, 0, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aelm.70580 by CochraneItalia, Wiley Online Library on [24/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

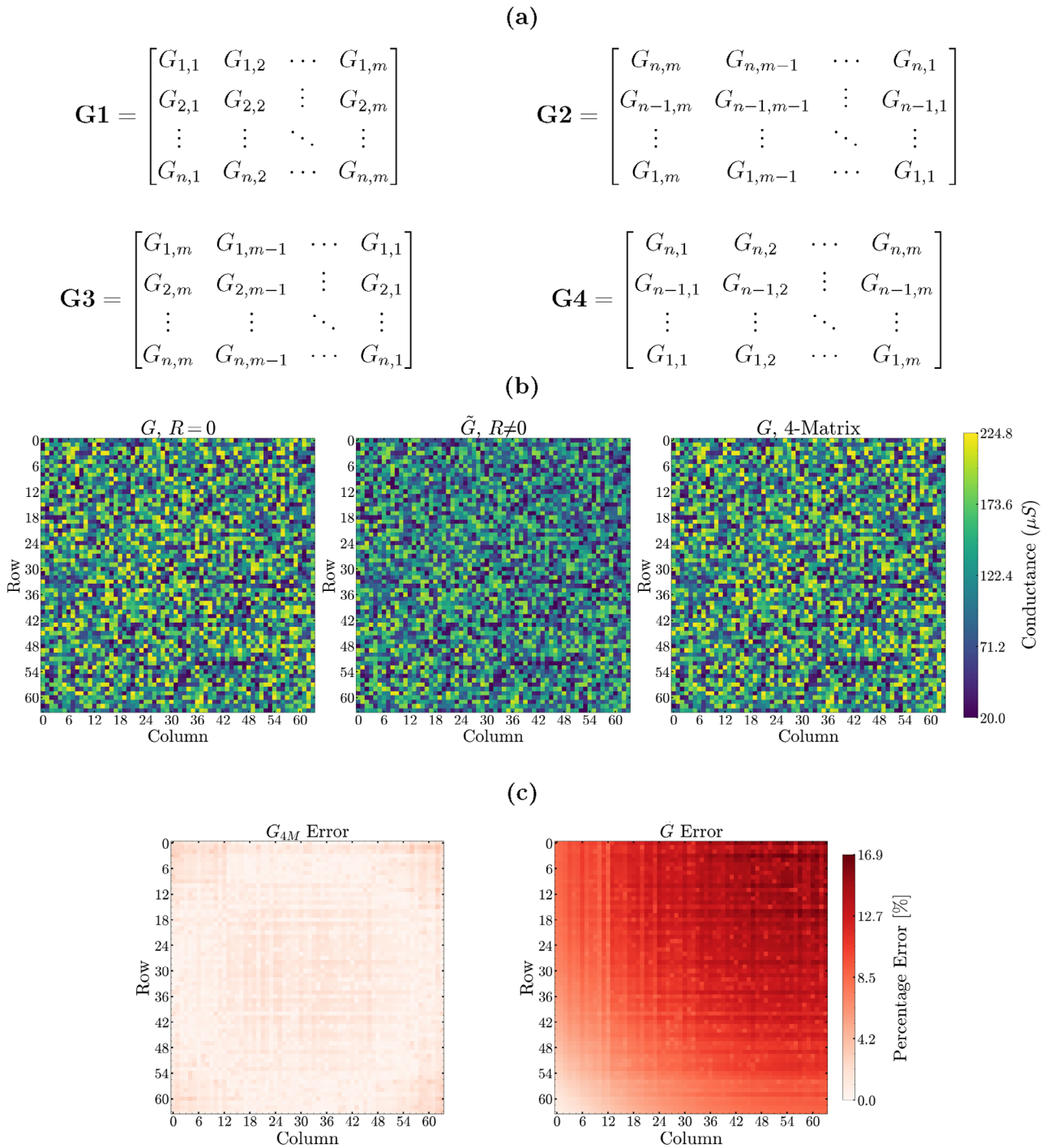

FIGURE 3 | | Conductance matrix transformations and crossbar effects. (a) Geometric and symmetric transformations of the four matrices **G1**, **G2**, **G3**, and **G4**. (b) Left panel: original conductance matrix **G** programmed in the crossbar. Middle panel: effective conductance matrix $\tilde{\mathbf{G}}$ including wire resistance effects. Right panel: corrected conductance matrix obtained after applying the four-matrix method. (c) Percentage error associated with each memconductance in the crossbar for $\mathbf{G_{4M}}$ and $\tilde{\mathbf{G}}$. Simulation parameters: $G_{\min} = 20\ \mu S$, $G_{\max} = 224.8\ \mu S$, $R = 0.35\ \Omega$, $n = 64$.

permutations with no matrix-specific recomputation. In this way, the correction burden is shifted from matrix-specific preprocessing and conductance reprogramming to a small number of analog MVM evaluations followed by output reordering and averaging.

This operation redistributes spatially localized deviations, such as voltage drops (conductance degradation) concentrated in specific crossbar regions, as illustrated in Figure 3, which tend to cancel out upon averaging. As a consequence, the overall output more closely approximates the ideal MVM, even in the presence of significant line resistance.

The computational overhead of this approach is modest: for the 64×64 arrays investigated in this work, the cost grows from one analog MVM to four transformed MVM evaluations, followed by

2199160x, 0, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aelm.70580 by CochraneItalia, Wiley Online Library on [24/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

output reordering, summation, and normalization. These four configurations provide sufficient spatial sampling to compensate the dominant row- and column-dependent IR-drop distortion in the demonstrated array size. For larger arrays, the same symmetry-averaging principle can be extended hierarchically by applying additional geometric transformations or partitions. In this generalized case, the number of required configurations is not strictly constant. For arrays larger than an elementary block of size b, validated up to $b = 64$ under the operating conditions used throughout this work (Supporting Information Section S13), the array can be partitioned into $(n/b)^2$ independently corrected blocks, each requiring the same four-configuration correction validated at $n = b$, so that total overhead grows as $4(n/b)^2$ for sequential block readout, while each individual operation remains a fully parallel analog MVM. In practice, the algorithm therefore preserves the intrinsic efficiency of in-memory computing while delivering a substantial improvement in accuracy. Simulation studies reported in this work confirm that, without correction, the error grows proportionally with the array size, quickly limiting scalability beyond 64×64. With the Four-Matrix Algorithm, by contrast, the error remains bounded, as observable in Figure 3, and exhibits a reduced dependence on array size.

2.5 | Hardware-Based DFT Calculations

To validate the generality and scalability of the proposed compensation scheme, we investigate a hardware-based discrete fourier transform (DFT) task implemented directly on the memristor crossbar array (see Section S8 for details). Because these measurements are obtained directly on fabricated hardware rather than through simulation, they inherently incorporate real device-to-device variability, programming errors, and read noise, rather than an idealized, noise-free conductance model. The DFT is a fundamental mathematical operation that converts a sequence of values from the time (or spatial) domain into the frequency domain, and it plays a central role in signal processing, image analysis, and computational physics [35, 36]. Importantly, the DFT can be expressed as a matrix–vector multiplication (MVM), where an $n \times n$ complex-valued transformation matrix multiplies a vector of dimension n. This formulation makes the DFT particularly suitable for hardware acceleration using memristor crossbar arrays, which natively perform MVM operations in parallel [5, 25, 37]. Because the hardware platform operates with real-valued conductances and input voltages, we program the real component of the normalized DFT matrix into a 64×64 crossbar tile. This choice preserves the orthogonality and frequency-selective properties of the transform while remaining compatible with single-ended analog conductance encoding. For each 64-dimensional input vector $\mathbf{x}$, the memristor crossbar performs the analog MVM in a single parallel operation:

$$\mathbf{i}_{\text{hw}} = \mathbf{G}_{\text{eff}}\mathbf{x}, \tag{8}$$

where $\mathbf{G}_{\text{eff}}$ denotes the effective conductance matrix including wire-resistance distortions. The resulting 64 column currents are digitized and compared with the ideal digital DFT output. In the absence of compensation, horizontal and vertical line resistances introduce position-dependent voltage drops that systematically distort the realized transform. Because IR-drop accumulates along the array, errors increase with array dimension, leading to structured deviations across output frequency bins. This scaling behavior severely limits accuracy for large crossbar implementations.

Figure 4 compares the ideal digital DFT output, the uncompensated analog output, and the compensated result. In the standard case, output errors increase systematically with array dimension due to cumulative interconnect losses. By contrast, the proposed compensation significantly reduces the column-wise percentage error and suppresses its scaling with array size. The corrected outputs closely track the ideal frequency components, demonstrating that wire resistance can be treated as a structured, compensable geometric perturbation rather than a fundamental scalability barrier.

2.6 | Multi-Layer Perceptron Results

To evaluate the effectiveness of the proposed Four-Matrix Algorithm, we adopt a system-level benchmarking approach based on a neural-network inference task, enabling a direct assessment of its impact on matrix–vector multiplication (MVM) accuracy in a realistic application scenario. While the MLP weights are programmed with realistic device noise via the AIHWKit Hardware Kit [21], this benchmark also reflects the noise-averaging property of the four-matrix algorithm derived in Section S12: the four independent programming instances required by the correction reduce the standard deviation of zero-mean programming noise by a factor of two relative to a single programming instance.

A *multi-layer perceptron* (MLP) architecture composed of three fully connected layers is considered. The network consists of an input layer with 256 neurons, followed by two hidden layers with 128 and 64 neurons, respectively, and a final output layer with 10 neurons, corresponding to the digit classes of the MNIST dataset. This configuration is intentionally selected to preserve a direct mapping between the network layers and MVM operations, while enabling the evaluation of error propagation across multiple analog computation stages. The network is evaluated using the MNIST handwritten digit dataset, a widely adopted benchmark for inference accuracy assessment. Each input image, resized to 16×16 pixels, is flattened into a 256-dimensional feature vector and processed sequentially through the three MVM operations associated with the fully connected layers. At each layer, the output activations are computed through analog or digital MVM, depending on the evaluation mode, and propagated to the subsequent layer. The final predicted digit label is obtained by selecting the class associated with the maximum activation at the output layer. Within this framework, MVM operations represent the dominant computational kernels, making the overall accuracy of inference highly sensitive to numerical errors introduced by analog hardware. This characteristic makes the MNIST classification task particularly suitable for assessing compensation strategies aimed at improving the fidelity of the analog MVM. Three computational modalities are compared for three different digits (complete digit set in Section S7, shown in Figures S1, S2, and S3) in Figure 5: ideal digital MVM, analog MVM without compensation, and analog MVM enhanced by the proposed Four-Matrix Algorithm. By analyzing the resulting class probability distributions and classification outcomes, we quantitatively demonstrate the ability of the

2199160x, 0, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aelm.70580 by Cochraneltalia, Wiley Online Library on [24/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

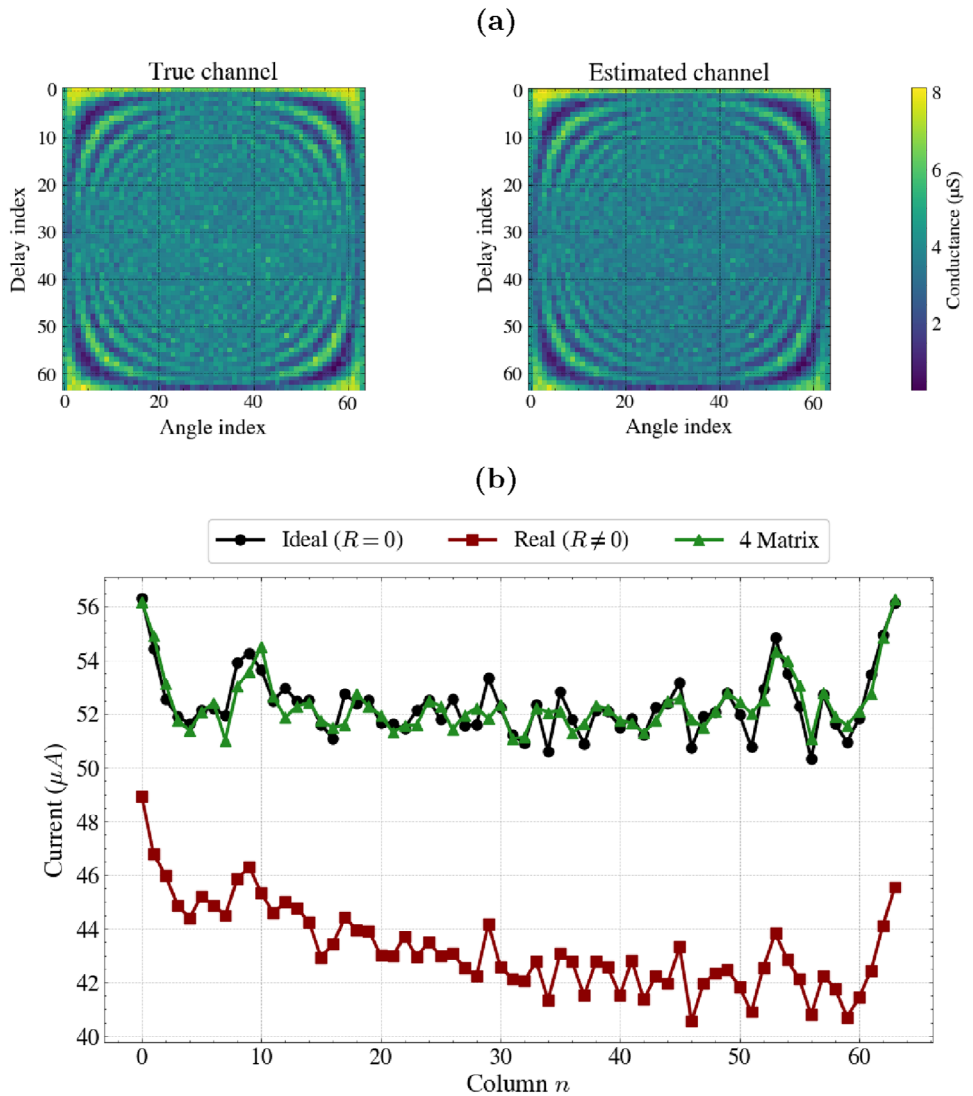

FIGURE 4 | DFT experimental implementation on the memristor crossbar. (a) Programmed real-valued 64×64 DFT conductance matrix and measured dot product engine platform (Section S8) effective conductance map showing spatial distortion induced by line resistance. (b) Column-wise output currents for the ideal digital DFT ($R = 0$), uncompensated hardware measurement ($R \neq 0$), and four-matrix compensated output. The compensation substantially reduces frequency-domain distortion and suppresses error scaling with array dimension on real hardware.

Four-Matrix Algorithm to mitigate analog computation errors and to limit error accumulation across multiple network layers. The results highlight the effectiveness of the proposed approach in restoring inference accuracy toward digital-level performance, thereby enabling reliable multi-layer neural-network inference in analog in-memory computing platforms.

2.7 | Hardware Cost and Accuracy Trade-Off

The four-configuration correction introduces an explicit trade-off between MVM accuracy, programming cost, read energy, latency, and hardware resources, as illustrated in Table 1. We distinguish here between the one-time cost required to program the conductance matrices and the per-input cost required to read them during inference. Let E_{prog} and T_{prog} denote the energy and latency required to program one conductance matrix, while E_{MVM} and T_{MVM} denote the read energy and read latency of one analog MVM after programming. The read latency includes array settling, sensing, and peripheral conversion. The additional cost of output reordering, summation, and normalization is denoted by E_{avg} and T_{avg}.

For the 64×64 arrays evaluated in this work, the corrected output is obtained from four transformed configurations. Therefore, the four matrices must be programmed before inference, with a setup cost of approximately $4E_{\text{prog}}$ and $4T_{\text{prog}}$. During inference, the per-input read-energy cost is approximately

$$E_{4\text{M}} \simeq 4E_{\text{MVM}} + E_{\text{avg}}. \tag{9}$$

The read-latency cost depends on how the four programmed configurations are accessed. If they are read sequentially using shared readout circuitry, the latency is

$$T_{4\text{M}}^{\text{seq}} \simeq 4T_{\text{MVM}} + T_{\text{avg}}. \tag{10}$$

2199160x, 0, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aelm.70580 by CochraneItalia, Wiley Online Library on [24/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

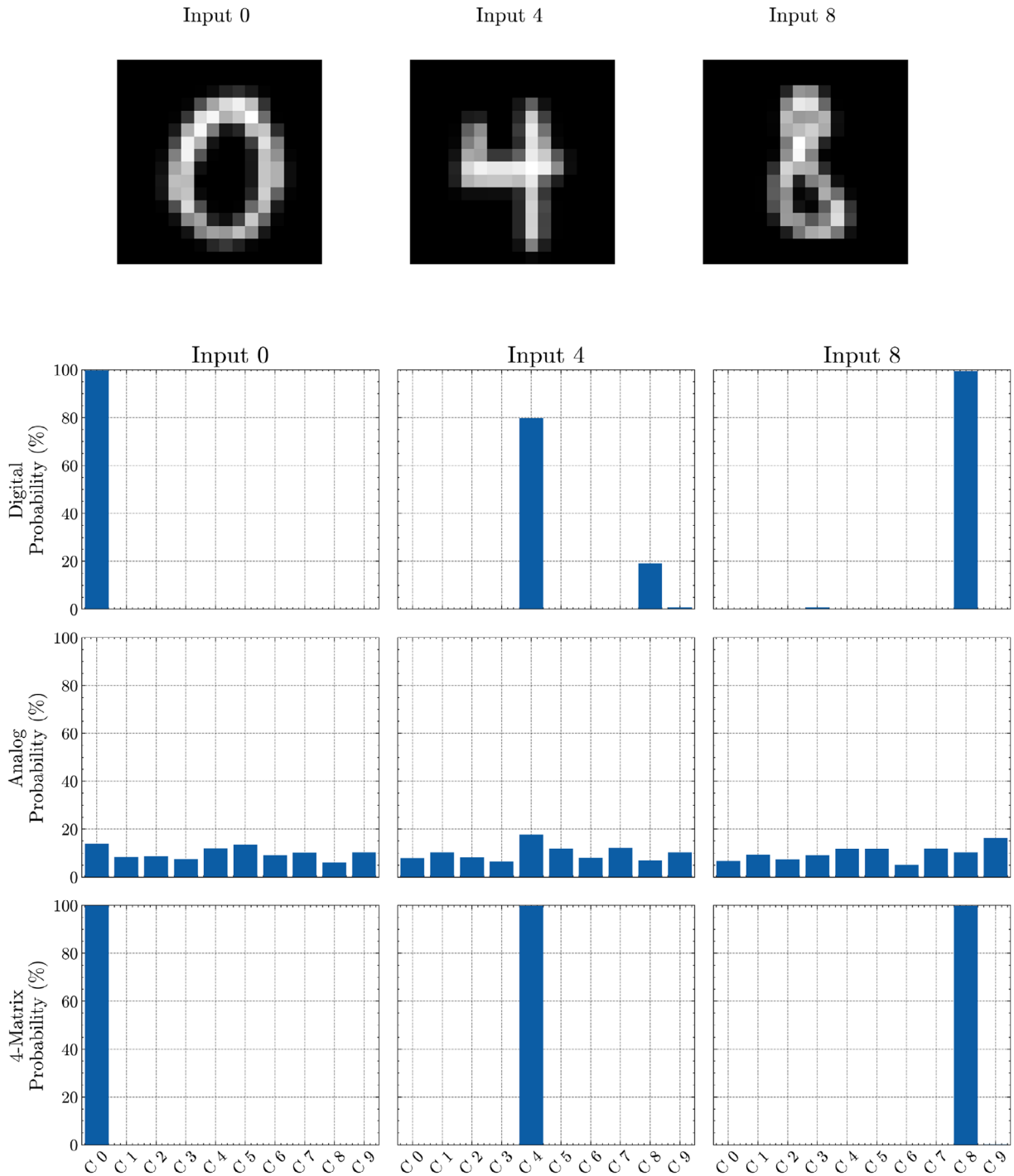

FIGURE 5 | MLP inference results with four-matrix compensation. (a) MNIST input images used for inference. Representative input samples corresponding to digits 0, 4, and 8, resized to 16×16 pixels and used as inputs to the multi-layer perceptron. (b) Corresponding output class probability distributions for digital, analog, and compensated analog MVM. Each column corresponds to a fixed input digit, while rows report the three computational modalities.

If they are read in parallel using parallel or sufficiently pipelined peripheral circuits, the latency approaches

$$T_{4M}^{\text{par}} \simeq T_{\text{MVM}} + T_{\text{avg}}. \tag{11}$$

Thus, the energy overhead is approximately proportional to the number of configurations, whereas the latency overhead depends on the degree of readout parallelism.

If a single physical array were reprogrammed for each transformed configuration and for each input, the programming latency and energy would dominate the computation. This is not the intended inference mode. Instead, the transformed matrices are programmed and stored before inference, and then reused over many input vectors. In this regime, the practical trade-off is between additional programmed array storage and read energy on the one hand, and improved MVM fidelity on the other.

2199160x, 0, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aelm.70580 by CochraneItalia, Wiley Online Library on [24/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 1 | Normalized hardware trade-off of the proposed correction. Energy and latency include both conductance programming and MVM readout.

Implementation	Energy cost	Latency cost	Hardware
Uncompensated MVM	$E_{\text{prog}} + E_{\text{MVM}}$	$T_{\text{prog}} + T_{\text{MVM}}$	1×
Sequential 4 MVM	$\sim 4E_{\text{prog}} + 4E_{\text{MVM}} + E_{\text{avg}}$	$\sim 4T_{\text{prog}} + 4T_{\text{MVM}} + T_{\text{avg}}$	1×
Parallel 4 MVM	$\sim 4E_{\text{prog}} + 4E_{\text{MVM}} + E_{\text{avg}}$	$\sim 4T_{\text{prog}} + T_{\text{MVM}} + T_{\text{avg}}$	4×
Hierarchical Extension	$\sim K(n)E_{\text{prog}} + K(n)E_{\text{MVM}} + E_{\text{avg}}$	$\sim K(n)T_{\text{prog}} + [T_{\text{MVM}}, K(n)T_{\text{MVM}}] + T_{\text{avg}}$	$K(n)$×

TABLE 2 | Accuracy-energy trade-off for MNIST classification.

Configuration	Accuracy	Energy
Digital (DRAM weight fetch)	81.01%	∼ 26 815.4 nJ
Analog, uncompensated (1 MVM)	15.83%	∼ 169.7 nJ
Four-Matrix (4 MVM)	79.23%	∼ 678.8 nJ

The benefit of this overhead must be evaluated against the uncompensated MVM error. In a single uncompensated crossbar, the wire-resistance error depends strongly on the array dimension, interconnect resistance, and selected conductance range. As the average relative MVM error increases, the analog output progressively loses numerical fidelity with respect to the intended matrix–vector multiplication. While the exact tolerance is task-dependent, errors of several percent are already problematic for precision-sensitive linear transforms and can also accumulate across multiple neural-network layers if not compensated or included during hardware-aware training. The four-configuration correction reduces this deterministic geometry-dependent error by averaging the row- and column-dependent distortion over complementary spatial configurations. Thus, the method trades additional programmed array storage and read energy for a substantial improvement in MVM fidelity, making the analog output usable in applications where the uncompensated crossbar error would otherwise dominate the computation.

2.7.1 | MNIST Classification: Accuracy and Energy

To complement the symbolic cost model above with a concrete estimate, we evaluate the accuracy-energy trade-off for the MNIST classification task described in Section 2.6. Accuracy is quantified as the mean probability assigned to the true class across the ten representative digits of Figure 5 and Section S7, yielding 81.01% for digital inference, 15.83% for uncompensated analog inference, and 79.23% for the Four-Matrix compensated case. Energy is estimated for one full forward pass of the network under a representative operating point, with full derivation given in Section S10; Table 2 summarizes the result.

The Four-Matrix Algorithm restores classification reliability close to the digital baseline while remaining approximately 40× more energy-efficient than digital inference under this representative estimate, and the 4× overhead relative to the uncompensated analog case leaves the method nearly two orders of magnitude below the digital energy cost.

TABLE 3 | Accuracy-energy trade-off for the hardware DFT experiment.

Configuration	NRMSE	Energy
Uncompensated (1 MVM)	17.51%	1.194 nJ
Four-Matrix (4 MVM)	2.50%	4.772 nJ

2.7.2 | DFT: NRMSE and Energy

The same accuracy-energy analysis applies to the hardware DFT experiment of Section 2.5. Under the same representative operating point adopted for the MNIST analysis above, the DFT hardware measurements show a normalized root mean square error (NRMSE) of 17.51% for the uncompensated case and 2.50% for the four-matrix-compensated case, a ∼ 7× reduction consistent with the improvement observed for MNIST classification, at the energy cost reported in Table 3. Full derivation is given in Section S10.

3 | Discussion

This work demonstrates that wire-resistance errors in memristive crossbars can be systematically suppressed by leveraging geometric symmetries of the conductance matrix. The resulting four-matrix algorithm mitigates cumulative distortions while preserving scalability, as confirmed by application-level benchmarks spanning both structured linear-algebra tasks and neural-network inference.

These experimental demonstrations were selected to validate the proposed correction on two representative MVM workloads. The 64×64 crossbar provides an intermediate-scale platform in which wire-resistance-induced spatial distortion is clearly observable while complete task-level operations can still be programmed and measured reliably. The DFT experiment evaluates a structured linear transform, where the numerical fidelity of the full MVM directly determines the output spectrum, whereas the MLP benchmark probes the propagation of compensated analog MVMs through multiple inference layers. These two cases therefore test complementary regimes of analog in-memory computing: precision-sensitive linear algebra and neural-network inference.

The framework itself is not restricted to this array size or conductance interval. At the simulation level, the same analytical model and correction principle can be extended to larger arrays,

2199160x, 0, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aelm.70580 by CochraneItalia, Wiley Online Library on [24/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

including 128 × 128 crossbars, by updating the correction parameters according to the array dimension, wire resistance, conductance range, and input-voltage range. More generally, the admissible conductance interval $[G_{\min}, G_{\max}]$ and the normalization factor $f(n, G_{\text{range}})$ are selected from the target effective precision, available device window, sensing resolution, and power budget. For the 64×64 arrays evaluated here, four geometric configurations are sufficient to suppress the dominant spatial distortion. For larger arrays, the same symmetry-averaging principle can be extended hierarchically by engineering additional geometric transformations or partitions, leading to a total overhead of $4(n/b)^2$ MVMs when the array is partitioned into blocks of the validated elementary size $b = 64$ (Section S13). Thus, different operating conditions primarily require re-parameterization of the analytical correction rather than a change in the underlying compensation principle.

Beyond qualitative trends, performance is conveniently reported in terms of effective digital precision. For a 64×64 array operated with $G \in [\,20\,\mu\text{S},\ 224.8\,\mu\text{S}\,]$, achieving approximately 10 effective bits requires $2^{10} = 1024$ discrete levels, which is compatible with recent advances in high-precision programming, redundancy-assisted protocols, and dynamic-precision operation [20, 33, 34]. Once the bit-precision target is set from the problem specification [8], one selects an admissible interval $[G_{\min}, G_{\max}]$ and computes a normalization factor

$$f = f\big(n,\, G_{\text{range}}\big), \quad G_{\text{range}} = [G_{\min}; G_{\max}],$$

that balances analog accuracy with resolution granularity. In hardware, the same factor rescales the aggregate response obtained by summing the four measurements associated with the transformed matrices, yielding a single corrected output without additional per-instance tuning. To cover signed linear algebra, differential conductance encoding maps each coefficient a_{ij} to a pair of programmed states,

$$a_{ij} \;\mapsto\; G_{ij}^{+} - G_{ij}^{-},$$

thereby enabling positive and negative entries and improving robustness to device-level variability [11, 23, 38]. For the 64×64 implementation demonstrated here, the computational workload is dominated by four vector–matrix multiplications. This four-configuration overhead should be understood as the operating point validated in this work rather than as a strictly size-independent requirement. Importantly, the demonstrated MLP benchmark highlights that the proposed framework is not limited to shallow architectures or single-layer mappings. Since the correction operates at the level of individual matrix–vector multiplications, it naturally extends to deeper neural networks and alternative architectures composed of cascaded linear layers, including fully connected networks of larger depth and width, convolutional layers mapped onto crossbar arrays, and hybrid neuromorphic models. As a result, the four-matrix algorithm provides a general and reusable strategy for improving inference reliability across a broad class of analog neural-network workloads.

The reliability improvement achieved by the Four-Matrix Algorithm originates from the suppression of deterministic, geometry-dependent errors associated with wire resistance and sneak paths. These non-idealities are set by the crossbar topology and can therefore be described analytically and mitigated through symmetry-based averaging. In practical hardware, additional effects such as device-to-device variability, programming errors, read noise, and temporal drift also contribute to the total MVM error budget. These mechanisms act differently from wire resistance: programming errors and device variability perturb the stored conductance matrix, read noise adds stochastic fluctuations to the measured currents, and temporal drift gradually changes the programmed conductance states over time [39, 40] Because the proposed correction averages complementary geometric configurations, random zero-mean noise and uncorrelated variability are not expected to be amplified and may be partially reduced across repeated or parallel evaluations. Systematic conductance drift, biased programming errors, or strongly correlated variability, instead, remain device-level effects that can be addressed through calibration, reprogramming, variability-aware mapping, or hardware-aware training [41, 42]. The proposed wire-resistance correction is therefore complementary to these established mitigation strategies and can be integrated with device-level calibration to improve robustness under realistic hardware conditions.

4 | Conclusion

This work shows that wire resistance in memristor crossbars does not constitute a fundamental scalability barrier, but rather a structured, geometry-induced bias that can be modeled and compensated analytically. By deriving a closed-form description of how horizontal and vertical interconnect resistances distort the effective conductance matrix, we demonstrate that their impact on analog MVM is predictable and, crucially, bounded.

Building on this insight, we introduced a correction strategy that operates entirely within the analog domain and preserves full parallelism. A first-order geometry-aware conductance adjustment captures the dominant wire-resistance effects, while a four-configuration symmetry averaging suppresses residual position-dependent errors without modifying peripheral circuitry.

Together, these steps suppress error accumulation with array size and maintain stable accuracy across realistic conductance windows. In the 64×64 arrays evaluated in this work, this correction is achieved using four geometric configurations, while larger arrays can be addressed by partitioning the array into independently corrected blocks of validated elementary size.

Beyond the specific correction scheme, the key contribution of this study is conceptual: it reframes wire resistance from an uncontrolled parasitic into a calibratable transformation whose structure is fixed by array geometry. Once the operating regime and conductance range are defined, correction parameters can be computed analytically and reused across runs and tasks, eliminating the need for iterative, task-specific calibration.

The present analysis is limited to quasi-static operation in the linear regime and assumes smoothly varying interconnect parameters. Extending the framework to distributed RC effects, mild device nonlinearities, and hardware-in-the-loop calibration on large, weakly non-uniform arrays represents a natural next

2199160x, 0, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aelm.70580 by Cochraneltalia, Wiley Online Library on [24/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

step. Nevertheless, the results already indicate that large-scale analog linear algebra can be made predictable, reusable, and scalable, provided that interconnect physics is treated as part of the computation rather than as an after-the-fact imperfection.

Acknowledgments

The authors would like to thank Gianluca Zoppo and Francesco Marrone, former Ph.D. students of the research group, for insightful discussions and contributions to the conceptual development of this work. The authors also acknowledge Sreetosh Goswami and Harivignesh S. for helpful discussions on the subject. This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under project CO 1916/1-1 "REservoir COMputing with MEmristive Nonlinear Dynamics: Theory, Design and Applications" (RECOMMEND).

Open access publishing facilitated by Politecnico di Torino, as part of the Wiley - CRUI-CARE agreement.

Conflicts of Interest

The authors declare no conflicts of interests.

Data Availability Statement

The data supporting the findings of this study are available from the corresponding author upon reasonable request.

References

1. D. Ielmini and H.-S. P. Wong, "In-Memory Computing with Resistive Switching Devices," *Nature electronics* 1, no. 6 (2018): 333–343.

2. W. Wan, R. Kubendran, C. Schaefer, et al., "A Compute-in-Memory Chip Based on Resistive Random-Access Memory," *Nature* 608, no. 7923 (2022): 504–512.

3. G. Migliato Marega, H. G. Ji, Z. Wang, et al., "A Large-Scale Integrated Vector–Matrix Multiplication Processor Based on Monolayer Molybdenum Disulfide Memories," *Nature Electronics* 6, no. 12 (2023): 991–998.

4. X. Zhang, A. Huang, Q. Hu, Z. Xiao, and P. K. Chu, "Neuromorphic Computing with Memristor Crossbar," *Physica Status Solidi (A)* 215, no. 13 (2018): 1700875.

5. M. A. Zidan, J. P. Strachan, and W. D. Lu, "The Future of Electronics Based on Memristive Systems," *Nature Electronics* 1, no. 1 (2018): 22–29.

6. Q. Xu, H. Geng, S. Chen, B. Yu, and F. Wu, "Memristive Crossbar Mapping for Neuromorphic Computing Systems on 3D IC," *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 25, no. 1 (2019): 1–19.

7. X. Wang, Y. Zhu, Z. Zhou, X. Chen, and X. Jia, "Memristor-Based Spiking Neuromorphic Systems Toward Brain-Inspired Perception and Computing," *Nanomaterials* 15, no. 14 (2025): 1130.

8. C. Li, M. Hu, Y. Li, et al., "Analogue Signal and Image Processing with Large Memristor Crossbars," *Nature Electronics* 1, no. 1 (2018): 52–59.

9. B. Feinberg, U. K. R. Vengalam, N. Whitehair, S. Wang, and E. Ipek, "Enabling Scientific Computing on Memristive Accelerators," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)* (IEEE, 2018), 367–382.

10. M. Hu, C. E. Graves, C. Li, et al., "Memristor-Based Analog Computation and Neural Network Classification with a Dot Product Engine," *Advanced Materials* 30, no. 9 (2018): 1705914.

11. H. Kim, M. Mahmoodi, H. Nili, and D. B. Strukov, "4K-Memristor Analog-Grade Passive Crossbar Circuit," *Nature Communications* 12, no. 1 (2021): 5198.

12. H. Bao, H. Zhou, J. Li, et al., "Toward Memristive In-Memory Computing: Principles and Applications," *Frontiers of Optoelectronics* 15, no. 1 (2022): 23.

13. G. Zoppo, A. Korkmaz, F. Marrone, et al., "A Mathematical Formulation of the Wire Resistance Problem in Memristor Crossbars," in *2022 IEEE 22nd International Conference on Nanotechnology (NANO)* (IEEE, 2022), 461–464.

14. L. Zhang, D. Borggreve, F. Vanselow, and R. Brederlow, "Impact of Parasitic Wire Resistance on Accuracy and Size of Resistive Crossbars," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE, 2021), 1–5.

15. R. Joshi and J. M. Acken, "Sneak Path Characterization in Memristor Crossbar Circuits," *International Journal of Electronics* 108, no. 8 (2021): 1255–1272.

16. E. Linn, R. Rosezin, C. Kügeler, and R. Waser, "Complementary Resistive Switches for Passive Nanocrossbar Memories," *Nature Materials* 9, no. 5 (2010): 403–406.

17. I. Vourkas, D. Stathis, G. C. Sirakoulis, and S. Hamdioui, "Alternative Architectures Toward Reliable Memristive Crossbar Memories," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 24, no. 1 (2015): 206–217.

18. S. Ngoc Truong, "Compensating Circuit to Reduce the Impact of Wire Resistance in a Memristor Crossbar-Based Perceptron Neural Network," *Micromachines* 10, no. 10 (2019): 671.

19. A. Levisse, B. Giraud, J.-P. Noël, M. Moreau, and J.-M. Portal, "Sneak-Path Compensation Circuit for Programming and Read Operations in RRAM-Based CrossPoint Architectures," in *2015 15th Non-Volatile Memory Technology Symposium (NVMTS)* (IEEE, 2015), 1–4.

20. W. Song, M. Rao, Y. Li, et al., "Programming Memristor Arrays with Arbitrarily High Precision for Analog Computing," *Science* 383, no. 6685 (2024): 903–910.

21. M. Rasch, T. Gokmen, D. Moreda, M. Le Gallo, and K. El Maghraoui, *IBM Analog Hardware Acceleration Kit*, (GitHub, 2023).

22. S. Ambrogio, P. Narayanan, H. Tsai, et al., "Equivalent-Accuracy Accelerated Neural-Network Training Using Analogue Memory," *Nature* 558, no. 7708 (2018): 60–67.

23. C. Li, D. Belkin, Y. Li, et al., "Efficient and Self-Adaptive In-Situ Learning in Multilayer Memristor Neural Networks," *Nature Communications* 9, no. 1 (2018): 2385.

24. A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory Devices and Applications for In-Memory Computing," *Nature Nanotechnology* 15, no. 7 (2020): 529–544.

25. M. Hu, J. P. Strachan, Z. Li, et al., "Dot-Product Engine for Neuromorphic Computing: Programming 1T1M Crossbar to Accelerate Matrix-Vector Multiplication," in *Proceedings of the 53rd Annual Design Automation Conference* (ACM Digital Library, 2016), 1–6.

26. A. Shafiee, A. Nag, N. Muralimanohar, et al., "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," *ACM SIGARCH Computer Architecture News* 44, no. 3 (2016): 14–26.

27. Z. Sun, G. Pedretti, E. Ambrosi, A. Bricalli, W. Wang, and D. Ielmini, "Solving Matrix Equations in One Step with Cross-Point Resistive Arrays," *Proceedings of the National Academy of Sciences* 116, no. 10 (2019): 4123–4128.

28. S. Ngoc Truong, "A Parasitic Resistance-Adapted Programming Scheme for Memristor Crossbar-Based Neuromorphic Computing Systems," *Materials* 12, no. 24 (2019): 4097.

29. N. Lepri, M. Baldo, P. Mannocci, A. Glukhov, V. Milo, and D. Ielmini, "Modeling and Compensation of IR Drop in Crosspoint Accelerators of Neural Networks," *IEEE Transactions on Electron Devices* 69, no. 3 (2022): 1575–1581.

2199160x, 0, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aelm.70580 by CochraneItalia, Wiley Online Library on [24/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

30. Y. Luo, S. Wang, P. Zuo, Z. Sun, and R. Huang, "Modeling and Mitigating the Interconnect Resistance Issue in Analog RRAM Matrix Computing Circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers* 69, no. 11 (2022): 4367–4380.

31. S. Gao, F. Yang, C. Lu, and Y. Zhao, "Fast Algorithms for Exact IR Drop De-Embedding in Analog Multiply–Accumulate Computing," *IEEE Transactions on Electron Devices* 69, no. 11 (2022): 6376–6383.

32. A. Agrawal, C. Lee, and K. Roy, "X-CHANGR: Changing Memristive Crossbar Mapping for Mitigating Line-Resistance Induced Accuracy Degradation in Deep Neural Networks," *arXiv* (2019): arXiv:1907.00285.

33. G. Pedretti, P. Mannocci, C. Li, Z. Sun, J. P. Strachan, and D. Ielmini, "Redundancy and Analog Slicing for Precise In-Memory Machine Learning—Part I: Programming Techniques," *IEEE Transactions on Electron Devices* 68, no. 9 (2021): 4373–4378.

34. S. Garg, J. Lou, A. Jain, Z. Guo, B. J. Shastri, and M. Nahmias, "Dynamic Precision Analog Computing for Neural Networks," *IEEE Journal of Selected Topics in Quantum Electronics* 29, no. 2: Optical Computing (2022): 1–12.

35. J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Mathematics of Computation* 19, no. 90 (1965): 297–301.

36. A. V. Oppenheim, *Discrete-Time Signal Processing* (Pearson Education India, 1999).

37. M. Prezioso, F. Merrikh-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* 521, no. 7550 (2015): 61–64.

38. Z. Liu, J. Tang, B. Gao, et al., "Neural Signal Analysis with Memristor Arrays Toward High-Efficiency Brain–Machine Interfaces," *Nature Communications* 11, no. 1 (2020): 4234.

39. G. Zoppo, A. Korkmaz, F. Marrone, S. Palermo, F. Corinto, and R. S. Williams, "Analog Solutions of Discrete Markov Chains via Memristor Crossbars," *IEEE Transactions on Circuits and Systems I: Regular Papers* 68, no. 12 (2021): 4910–4923, https://doi.org/10.1109/TCSI.2021.3126477.

40. G. Zoppo, "Dynamic Neural Networks and Brain-Inspired Computing," (Ph.D. thesis, Ph.D. in Electrical, Electronics and Communications Engineering, Politecnico di Torino, 2022).

41. A. P. James and L. O. Chua, "Variability-Aware Memristive Crossbars—A Tutorial," *IEEE Transactions on Circuits and Systems II: Express Briefs* 69, no. 6 (2022): 2570–2574.

42. M. J. Rasch, C. Mackin, M. Le Gallo, et al., "Hardware-Aware Training for Large-Scale and Diverse Deep Learning Inference Workloads Using In-Memory Computing-Based Accelerators," *Nature Communications* 14, no. 1 (2023): 5282.

Supporting Information

Additional supporting information can be found online in the Supporting Information section.

Supporting File: aelm70580-sup-0001-SuppMat.pdf.

2199160x, 0, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aelm.70580 by CochraneItalia, Wiley Online Library on [24/09/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License