

RESEARCH ARTICLE OPEN ACCESS

Highly-Uniform Passive Crossbar Arrays of Resistive Switching Random Access Memory (RRAM) for In-Memory Computing Applications

S. Ricci | P. Mannocci | M. Porzani | D. Bridarolli | F. Carletti | M. Farronato | D. Ielmini 

Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano and IU.NET, Milano, Italy

Correspondence: D. Ielmini (daniele.ielmini@polimi.it)

Received: 27 January 2026 | Revised: 23 May 2026 | Accepted: 15 July 2026

Keywords: artificial intelligence | crossbar array | in-memory computing | neural networks | quadratic assignment problem | Resistive switching memory (RRAM)

ABSTRACT

Resistive switching random-access memory (RRAM) is gaining attention for its ability to support various data-intensive computing tasks via in-memory computing (IMC). The use of passive RRAM arrays is hindered by sneak-path issues, programmability challenges, and concerns regarding reliability and robustness. In this study, we address the initialization problem and propose a strategy to efficiently program passive arrays, thereby overcoming these limitations. We validate our approach using 32×32 crossbar arrays of Pt/HfO₂/Ti RRAM, achieving 99.5% functionality and encoding various conductance weights with an error below 3%. To demonstrate the potential of RRAMs, we experimentally perform 24×24 discrete cosine transform (DCT) image reconstruction, achieving an accuracy higher than 95%. Also, we implement a single-layer classifier for feature recognition on the MNIST digit dataset, achieving 84.6% accuracy compared to the theoretical 86.1%. We finally develop an in-memory solver for combinatorial optimization problems, such as the quadratic assignment problem, leveraging the full parallelism of the crossbar configuration and implementing a quantum-inspired parallel annealing method, and demonstrate non-linear regression up to the fourth order.

1 | Introduction

Data-intensive workloads such as artificial intelligence (AI), augmented reality, and generative models face escalating demands for computational power, faster data processing, and reduced energy consumption. To meet these challenges, in-memory computing (IMC) has emerged as a new paradigm enabling the execution of computation and information storage within the same memory unit [1, 2]. Several efforts have been reported in the literature to implement IMC architectures benefiting from the technological advancement and robustness of traditional memories such as static random-access memory (SRAM) [3] and dynamic-random access memory (DRAM) [4].

On the other hand, emerging nonvolatile memories such as phase-change memory (PCM) [5–7], magnetic random access memory (MRAM) [8, 9], electrochemical RAM (ECRAM) [10–13], conductive-bridge (CBRAM) and resistive RAM (RRAM) [14–21] have been proposed as a valid support capable of accelerating matrix-vector multiplication (MVM), which provides the backbone of essential applications such as deep neural network (DNN) and discrete Fourier transform (DFT). This paradigm shift is enabled by the parallelism of the crossbar memory architectures, leveraging Kirchhoff's and Ohm's laws to carry out an analog MVM between the input voltages and the conductance matrices [1].

This is an open access article under the terms of the [Creative Commons Attribution](https://creativecommons.org/licenses/by/4.0/) 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

Among the various memory technologies, RRAMs show the most promising properties, thanks to the simple and versatile fabrication process flow, the small device-to-device variability, the good retention and the ability to store multilevel states. Current RRAM technology predominantly relies on the one-transistor/one-resistor (1T1R) structure [22, 23], where each memory cell is combined with a selection transistor, which is responsible for the correct programming and selection preventing unwanted interference between crossbar devices [24]. The 1T1R configuration ensures precise programming and eliminates device interference on the same wordline or bitline. The correct operation of crossbar architectures during programming relies on the possibility of writing the individual memory elements without disturbing the others, since the voltages are applied simultaneously to all devices of the wordlines or bitlines [25].

On the other hand, the select transistor negatively impacts on the cell footprint and circuit complexity, as well as making 3D integration of crossbar arrays difficult to achieve [26]. Achieving reliable passive crossbars [27–29] might speed up the scaling and 3D integration of RRAM for energy-efficient accelerators based on IMC.

In this study, we aim to advance the implementation of passive crossbar arrays of RRAM by experimentally demonstrating their high accuracy in various applications based on MVM, as illustrated in Figure 1a. We demonstrate a 32×32 passive crossbar array of RRAM based on a Pt/HfO₂/Ti stack, achieving 99.5% yield, namely, only five cells do not show repeatable switching. We first show proper device initialization and perform basic characterization to examine device switching and variability within the array. We then evaluate the response to pulse programming within a program-and-verify (P&V) approach. To address the issue of sneak paths, we propose a methodology where the crossbar array is pre-programmed to a uniform conductance level above the target level, then each device is gradually reset to the desired state by maintaining a total relative error of less than 3% with respect to the target. Finally, we validate our programming strategy by bringing different algorithms to the analog domain and solving the related task with an average accuracy higher than 96%. State-of-the-art processing workloads, such as image compression based on the discrete cosine transform (DCT) algorithm, a single-layer neural network classifier, a combinatorial logic solver, and a non-linear regression accelerator, are demonstrated with our crossbar array, thus confirming the feasibility of IMC with passive arrays of relatively large size.

2 | Results

2.1 | Device Initialization and Basic Characterization

The 32×32 crossbar arrays in this study were fabricated by standard CMOS processes to ensure compatibility with back-end-of-line (BEOL) integration. Figure S1 shows the details on the fabrication process flow. The RRAM structure consists of a metal-insulator-metal (MIM) stack [30–32] with an oxide layer sandwiched between the top electrode (TE) and the bottom electrode (BE) [33], as depicted in Figure 1b. To perform the reading and programming of each individual device within the

crossbar array, we developed the custom printed circuit board (PCB) presented in Figure S2. Figure 1c shows a scanning electron microscopy (SEM) image of the 32×32 array used in this work. The RRAM devices are located at the intersections of the BEs and TEs, as illustrated in Figure 1d, where vias are etched through the oxide spacer. The pristine conductance values of the array are reported in the conductance map in Figure S3. Over 90% of the devices are in a high-resistance state (HRS) with conductance below 100 nS, indicating that a forming operation is necessary to initialize these devices. The remaining devices are in a low-resistance state (LRS), with conductance ranging from 1 μ S to 1 mS, likely due to fabrication imperfections.

As an initial step, a reset operation was applied to all RRAM devices to decrease the conductance below 1 μ S, which we take as a reference for the threshold between HRS and LRS. The forming operation is then carried out on all devices, starting with those in deeper HRS values that require higher voltages, followed by resetting the devices back to HRS. Figure 1e and Figure S4 show the forming I–V curves for all RRAM devices. The forming map in Figure S5a presents a clear area where voltages have larger values with respect to the average, which might be due to the already-initialized devices, which are involuntarily switching during other forming operations. In fact, the current sunk by cells in the LRS may cause IR drop, thus increasing the required voltage to induce forming of other cells in the same current path. After forming, few cycles of set and reset processes are applied to each device to assess the correct switching operation. Figure 1f reports 10^3 set/reset cycles of a device inside the array, while Figure S6 reports the statistics for all the devices. During the set/reset operation, all the other devices on the same BE and TE are monitored to prevent unintentional switching. Set voltages are in the range between 0.5 and 2 V while reset voltages are in the range between –2 and –1 V, as reported in Figure S5. We found no correlation between the set and reset voltages in each device, as illustrated in Figure S7d. Figure 1g shows the probability distributions of the LRS conductance for all the devices of the array after set operation with a compliance current of 600 μ A, indicating a log-normal distribution. Finally, Figure 1h shows the measured low-voltage I–V curves for the LRS, indicating a strong linearity of the RRAMs. Good linearity is indeed essential to implement precise multiplication according to the Ohm's law $I = G \cdot V$. Note that devices start to show non-negligible non-linearity above 150 mV, as reported in Figure S8. Memory cells in the LRS were also monitored for at least 12 h to ensure stability and negligible drift, as reported in Figure S9.

2.2 | Multilevel Programming Methodology

We studied multilevel programming of RRAM devices by using square pulses of appropriate amplitude and duration. The current was limited by an external compliance or a series transistor during the set transition. No current limitation was adopted during the reset operations, with sneak-path issues being experimentally less significant [34]. Figure 2a shows the device conductance as a function of the number of reset pulses at constant amplitude. The device was initially programmed to an LRS conductance $G = 500 \mu$ S, then reset pulses of pulse width 10 μ s were sequentially applied while the conductance was measured at 100 mV after each reset pulse. As the pulse amplitude V_p

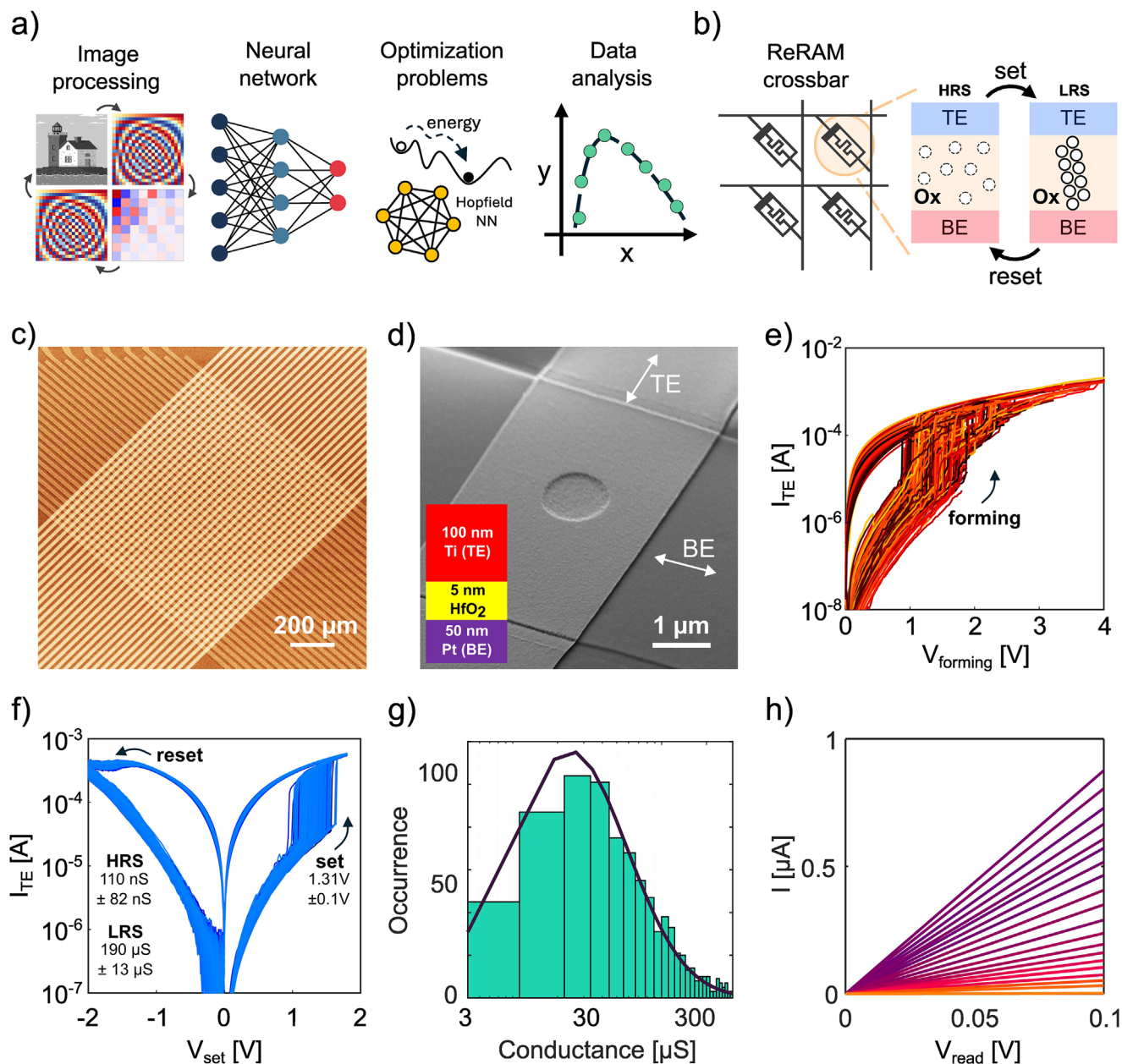


FIGURE 1 | 32 × 32 passive array based on Pt/HfO₂/Ti RRAMs: characterization, initialization, and cycling performances. (a) Examples of applications benefiting from RRAM-based IMC architecture, where the massive use of MVM operations can be carried out in the analog domain by storing the weights inside the RRAM. (b) Working principle of a RRAM and its internal configuration. Set and reset operation are responsible for the conductive channel established between the TE and BE, and thus the conductance of the memory can be tuned from a HRS to a suitable LRS. (c) SEM image of a 32 × 32 crossbar array, where the memories are hosted in vias at the intersection of the TEs and BEs. (d) Detail of a via hosting the RRAM memory and the corresponding stack. Vias isolate the memory location and reduce the device-to-device variability. (e) Initialization of the devices by forming I-V curves. The image collects the first positive cycle of all devices, with only 5 unswitched devices out of 1024, equivalent to 99.5% working devices. (f) Example of 1000 set and reset operations. (g) Distribution of the LRS state reached from all the devices after setting with the proper voltage and 600 μA compliance current. (h) Low voltage I-V sweep to check linearity for random states.

increases above −0.8 V, the conductance gradually decreases, with higher V_p leading to more effective resets with fewer pulses. Note that a conductance saturation occurs after some pulses, indicating that the reset amplitude becomes no longer adequate to reset the device. As the pulse duration increases, the reset operation leads to a deeper saturation point with a lower G , as shown in Figure S10a. An alternative reset methodology involves increasing V_p at each pulse to prevent the saturation effect. This

is shown in Figure 2b, where G is reported as a function of the number of pulses, starting from different conductance states and leading to a fully reset operation in the HRS. The most suitable V_p depends on the pulse duration, namely, relatively low voltages are sufficient for relatively long durations (e.g., from 0.8 to 1.3 V at 1 ms) while relatively high voltages are adequate for relatively short pulses (2 to 2.5 V @ 1 μs), as shown in the Figure S10b. Figure 2c shows the measured G for increasing

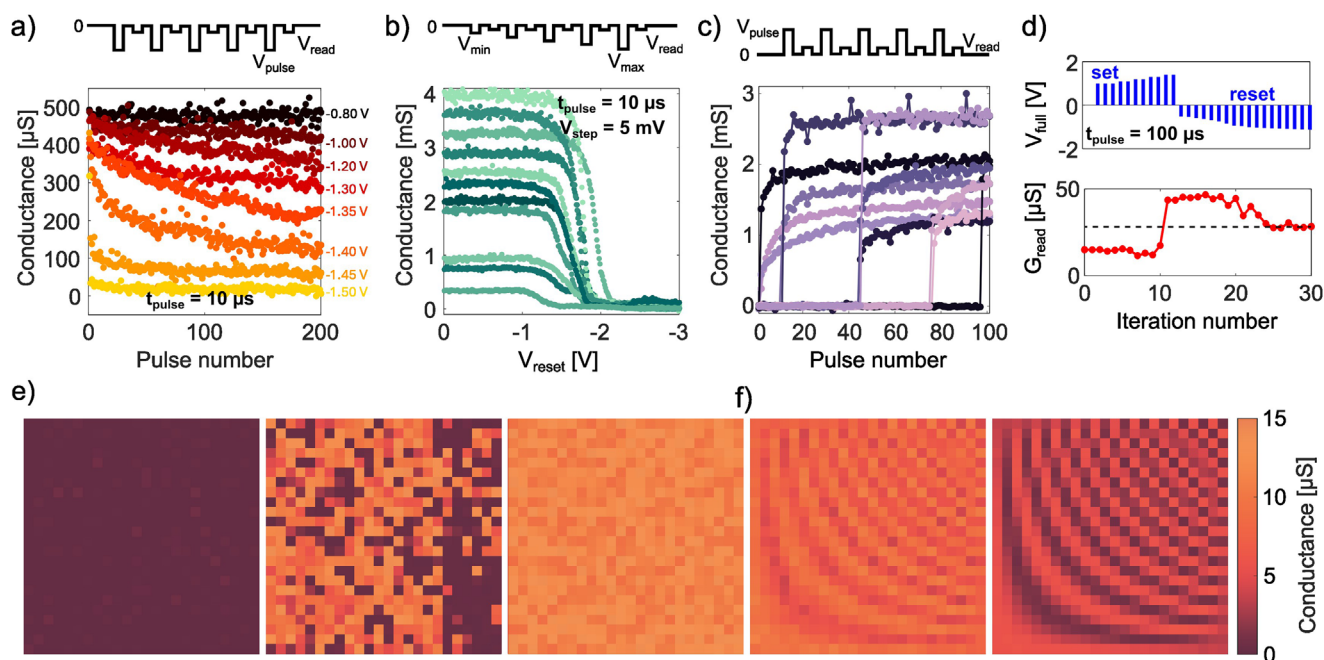


FIGURE 2 | Characterization of the device response to programming pulses, program and verify examples, and proposed strategy to program passive arrays. (a) Conductance tuning using constant pulses by varying amplitude and pulse duration, starting from 500 μS . The conductance is monitored through a 0.1 V reading pulse between each reset pulse. With $V_{\text{reset}} = -0.8$ V the conductance is not changing even after many reset pulses. Increasing the amplitude makes the reset process more evident until the first pulse is enough to almost completely reset the device ($V_{\text{reset}} = -1.5$ V). (b) Increasing reset pulse amplitude response. In green, the different starting conductance levels with 10 μs pulses and 5 mV step. (c) Set operation using constant pulses of 100 μs pulses with 0.9–1.6 V amplitude. Set transition does not show a clear trend with the amplitude and the duration. (d) Example of program and verify operations for a device that quickly reached the target. Positive pulses set the conductance to higher values while negative ones reduce the values, acting as a reset. (e) Proposed strategy to program passive arrays. Since some devices show instabilities and sneak-path issues, the overall matrix is programmed at a uniform conductance value above the target, for example, between 8 and 20 μS when we want to program in a 0–5 μS . It should be noted that the unwatchable devices have been reallocated by swapping rows and columns but they can potentially encode an HRS state. During the set, other half-selected devices already in LRS can switch, thus it is fundamental to properly select the range to allow all the devices to be programmed. (f) Reset operations involve lower voltage amplitudes compared to the set operation, thus there are fewer sneak-path issues. The target is programmed by only resetting the devices, approaching the values by reducing the distance and the programming step at each iteration.

number of pulses at constant amplitude. Contrary to the reset operation, the conductance increases abruptly from the HRS after the application of a random number of pulses, reflecting the formation of a conductive filament. By combining the speed of the set transition with the precision of the reset operation, conductance can be efficiently tuned by accurate P&V algorithms. This is shown in Figure 2d, reporting the measured G during the P&V process, including a number of set pulses to increase the conductance above the target point, followed by a number of reset pulses which allow to minimize the error between the final G and the target state.

The application of the P&V algorithm to a device within the crossbar array may lead to parasitic disturbance of devices in the selected row or column, during both set and reset processes. To prevent possible parasitic disturbances, we optimized the P&V algorithm according to the following 2-step methodology: (i) all devices in the array are first programmed above a given conductance, and (ii) they are then gradually reset to the target conductance. Figure 2e,f illustrates the map of conductance during the P&V algorithm for a 24×24 subarray of the crossbar circuit. First, all RRAM devices are programmed from the HRS to an LRS conductance range between 8 and 15 μS . Then, a gradual reset operation is carried out to decrease the conductance to the

target value. In the figure, we considered a 24×24 discrete cosine transform (DCT) matrix as a target conductance. Figures S11 and S12 provide more details about the first and second steps, respectively. Figure S13 shows the average number of reset pulses required to tune the matrix, considering all the iterations. Figure S14 compares the target conductance matrix with the experimental result after programming. The total programming error, expressed as the ℓ_2 norm, is 3.1%, with a standard deviation $\sigma = 97$ nS.

Notably, RRAM programming is typically more power- and latency-hungry than conventional volatile memories such as static random access memory (SRAM) or dynamic RAM (DRAM) [35]. However, (i) the programming overhead can be adjusted depending on the required weight mapping accuracy of the target application, and (ii) the weight-stationary nature of typical AI tasks [36–38], together with the nonvolatile characteristics of RRAM, allows programming to be executed *una tantum*, and amortized across multiple computations.

On the other hand, due to the absence of a selecting element, IR drop can ultimately limit the scaling of the proposed programming method to larger memory arrays. To enable scaling to larger array sizes, several directions can be explored including

(i) controlling the device LRS, thereby limiting the current flowing through selected lines, (ii) increasing the conductivity or width of metal interconnects, and (iii) using parallel or shorted lines to effectively reduce line resistance, with predicted maximal crossbar sizes in the order of 64×64 to 128×128 . To further analyze the scalability of the proposed approach, we performed circuit-level simulations considering array sizes from 2×2 to 128×128 , line resistances equal to 4Ω [39], and LRS conductances from $1 \mu\text{S}$ to $25 \mu\text{S}$ (Extended Data Figure S15). Consistently with previous analytical models [40–42], the IR-drop-induced conductance deviation follows an approximate $\mathcal{O}(N^2 \cdot r \cdot G_{\text{LRS}})$ scaling. Under a target effective conductance error of 3%, the analysis suggests LRS conductances of approximately $15 \mu\text{S}$, $5 \mu\text{S}$, and $1 \mu\text{S}$ for 32×32 , 64×64 , and 128×128 arrays, respectively, for $r \simeq 4 \Omega$. These results provide practical design guidelines for selecting the LRS range as a function of array size and target accuracy [43]. Furthermore, to enable the solution of larger-scale problems, modular architectures comprising multiple crossbars with shared programming peripherals will be required [44–46], with arrays operating in parallel to preserve the latency advantage of IMC over classical computers.

3 | IMC Applications

To fully support the RRAM crossbar array and multilevel programming methodology for IMC, we considered typical data-intensive tasks relying on massive MVM, namely (i) image processing with DCT, (ii) neural networks, (iii) quadratic unconstrained binary optimization (QUBO) and (iv) non-linear regression.

3.1 | High-Performance Image Compression With DCT

Fourier transform (FT) is an essential numerical tool that is generally adopted for image processing and compression via image decomposition into its sine and cosine components [47]. The FT output represents the image in the frequency domain, similar to the case of one-dimensional time series data. The DCT provides a discrete version of the FT, which executes image decomposition via the cosine function only. An image can be thus described as the superposition of a finite set of functions referred to as the *basis* functions, which are universal, and weighted through proper coefficients. The size of the DCT depends on the number of used basis, or spatial wavelengths, and dictates the fidelity of image decomposition. For our scope, we selected a video stream collected by a dashcam with 1920×1080 resolution at a frame rate of 60 Hz. Frames are then extracted and rescaled to a 192×360 size to demonstrate the IMC-based DCT and spatial reconstruction. Figure 3a shows a sketch of the working principle of the algorithm, referring to the case of a traffic light image, where the 24×24 DCT is applied across the image. The selected detail of 24×24 size can be seen as the superposition of many different basis functions.

We adopted a 24×24 DCT size and a reference row/column for sign representation in the RRAM array. Among the various DCT algorithms, we selected the DCT-IV form, whose transform matrix $\mathbf{W}$ is orthogonal (i.e., $\mathbf{W} = \mathbf{W}^T$). The symmetric form

allows for a simple and straightforward computation. In fact, from a mathematical point of view, the frequency coefficients $\mathbf{Y}$ of the input image $\mathbf{X}$ are extracted from the equation $\mathbf{Y} = \mathbf{W}\mathbf{X}\mathbf{W}^T$, which thus becomes $\mathbf{Y} = \mathbf{W}\mathbf{X}\mathbf{W}$ in the case of DCT-IV. Image compression is achieved by retaining coefficients of $\mathbf{Y}$ above a given threshold, typically resulting in annihilation of high-frequency components. Image reconstruction from the compressed frequency coefficients $\tilde{\mathbf{Y}}$ then follows the equation $\tilde{\mathbf{X}} = \mathbf{W}^T \tilde{\mathbf{Y}} \mathbf{W}$, which can be rewritten as $\tilde{\mathbf{X}} = \mathbf{W} \tilde{\mathbf{Y}} \mathbf{W}$ for DCT-IV. The pixel depth is 256 levels (8-bit) grayscale colors, which are encoded in the applied voltage range between 0 and 100 mV. Figure 3b and Figure S16 compare the original and the reconstructed image, where the matrix-matrix multiplications are carried out via iterated MVMs in the analog domain with our crossbar array. Owing to limitations of our experimental setup, characterized by a single-channel DAC (see Extended Data Figure S2), we perform multiplication *ex situ* in the digital domain. The reconstructed image shows high quality and accuracy with respect to the original image, with errors mainly occurring within areas of constant or slowly varying color, such as in the sky. Figure 3c shows the correlation plot of the reconstructed image as a function of the input image, highlighting excellent correlation. To measure the accuracy of our system, we calculate the peak signal-to-noise ratio (PSNR) [48], namely:

$$\text{PSNR} = 10 \log_{10} \frac{\text{MAX}_I^2}{\text{MSE}} \quad (1)$$

where MAX_I is the maximum pixel value and MSE is the mean squared pixel reconstruction error. From our experimental data, we obtain a PSNR of 32 dB, equivalent to a 4.25-bit fully-digital reconstruction. Figure 3d provides more information about the MVM accuracy, reporting the correlation plot for each step of the computation and evidencing the gradual accumulation of experimental error. The latter can be attributed to the non-ideal mapping of the DCT coefficients in the RRAM array and experimental noise.

3.2 | Neural Network

In this experiment, we focus on a simple neural network consisting of a single-layer perceptron to support the RRAM for AI applications [49–51]. Figure 4a shows a conceptual illustration of the neural network for handwritten digit recognition [52, 53]. The input signals, consisting of the image pixel amplitude, are processed via the weighting matrix $\mathbf{W}$ mapped in the crossbar array. The activation function at the output neuron is responsible for the classification, where only the neuron associated with the specific digit is activated. The dataset contains 28×28 grayscale (256 levels, 8-bit) images, resized in 10×9 with the *bilinear* method to maximize the matrix size that can be encoded in the passive array. Figure 4b shows the 90×10 weighting matrix (90 inputs, 10 outputs) that was trained in software by using the 60 000 training images with a learning rate $\eta = 0.01$ and batch size 10. Further details of the adopted perceptron model and training are reported in the Methods section. The weight matrix was sequenced into three blocks and mapped in the crossbar array via accurate P&V algorithm, as shown in Figure 4b. The MVM is computed by summing the currents of 3 analog MVMs in the

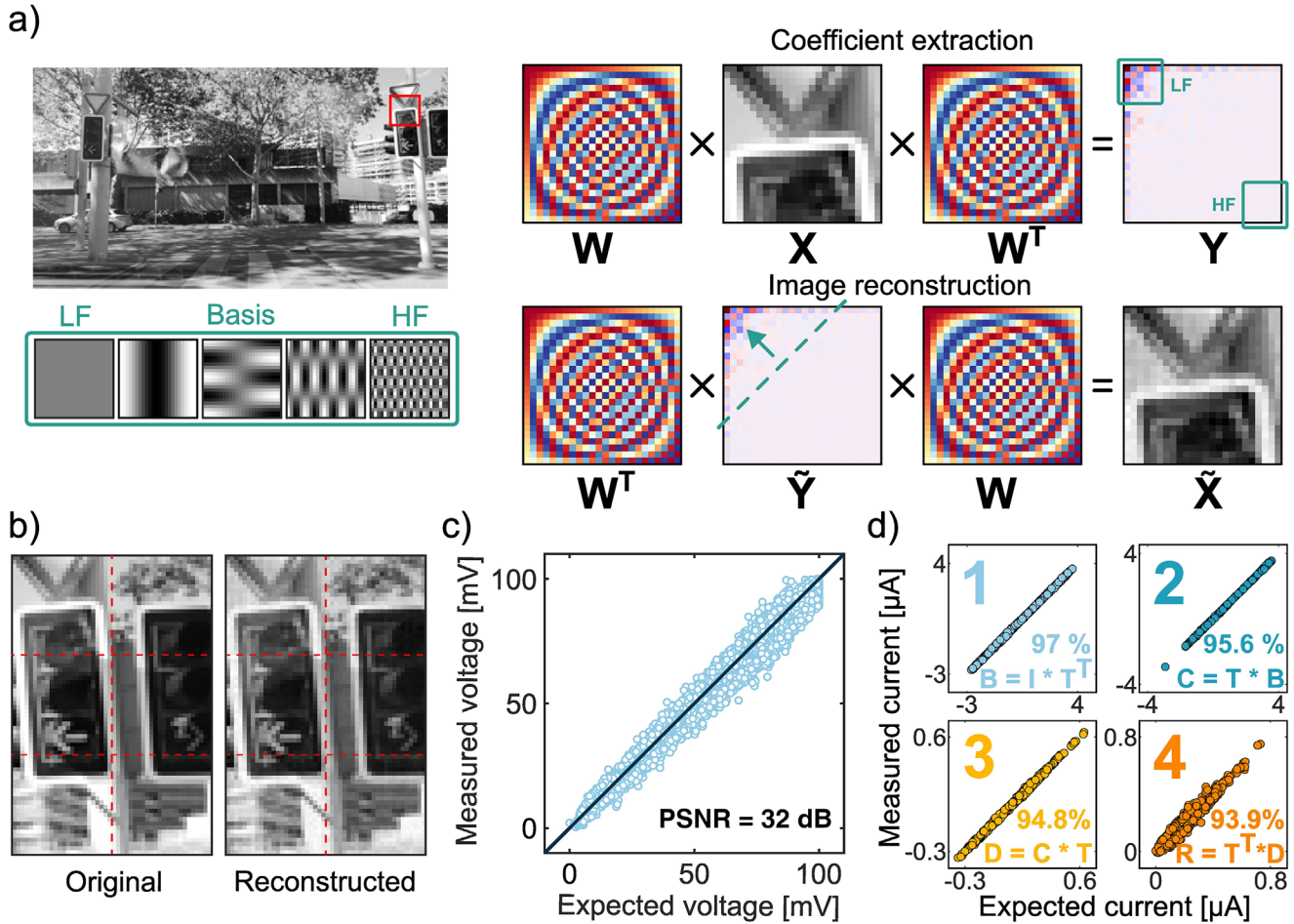


FIGURE 3 | Demonstration of spatial Discrete Cosine Transform (DCT) algorithm applied to image reconstruction. (a) Sketch and working principle of the image compression and reconstruction algorithm, where an image is decomposed as a sum of a set of properly weighted basis functions defined by spatial wavelengths. The algorithm consists of two steps, namely coefficient extraction and reconstruction. Coefficients (Y) are first extracted by performing blockwise matrix-matrix product between the image (X), the DCT matrix (W), and its transpose (W^T), typically yielding highly sparse Y matrices, with information concentrated in the low-frequency regime (top left). To perform compression, high-frequency components are typically discarded using a suitable filter or mask, obtaining a compressed spectrum $\tilde{Y}$. The image can finally be reconstructed ($\tilde{X}$) by summing the weighted basis function or by applying the DCT transformation. (b) Comparison between details of the reconstructed and the original image. (c) Correlation plot of all the pixels experimentally reconstructed and the analytical solution. Pixels are rescaled from 0–255 levels to 0–0.1 V. For each block, values are rescaled to minimize the error. (d) Correlation plot of all the currents during the 4 matrix-matrix products of the algorithm, highlighting accumulation of programming errors across the various algorithm steps.

digital domain. Figure 4c shows a correlation of the measured MVM currents as a function of the ideal values for the whole test set, consisting of 1000 images. A winner-take-all (WTA) activation function was adopted, where the output of the neuron with the largest current was set to 1, while all other neurons were set to 0. Figure 4d shows the accuracy obtained during training as a function of the bit precision of the weights, compared to the ideal accuracy of 86.1% with floating point 64-bit (FP64) precision. The IMC-based inference achieved an accuracy of 84.6%, where the slight decrease from the software level can be attributed to the RRAM conductance error. Comparison with results in Figure 4d indicates that the RRAM precision is equivalent to 5 bit, in line with the mapping error around 3% precision of the P&V algorithm for multilevel programming. Figure 4e reports the confusion plot, indicating that the perceptron classification has a good accuracy with only minor confusion among digits.

3.3 | QUBO Solver With Quantum-Inspired Annealing

The RRAM crossbar array was adopted for an IMC-based implementation of a quantum-inspired parallel annealing implementation of a QUBO solver. QUBO is a crucial computational tool to solve a wide range of problems, including scheduling, logistics and circuit/network design via Hopfield neural network (HNN) [54]. In general, these problems aim at minimizing a cost function where a large number of yes/no decisions must be made [55]. In quantum-inspired annealing, the system is initially prepared in a reference state, then it is allowed to evolve freely, ideally converging to the global minimum, as sketched in Figure 5a. A common issue in these approaches is the system getting stuck to a local energy minimum. Escape from local minima can be supported by adding intentional noise, where the

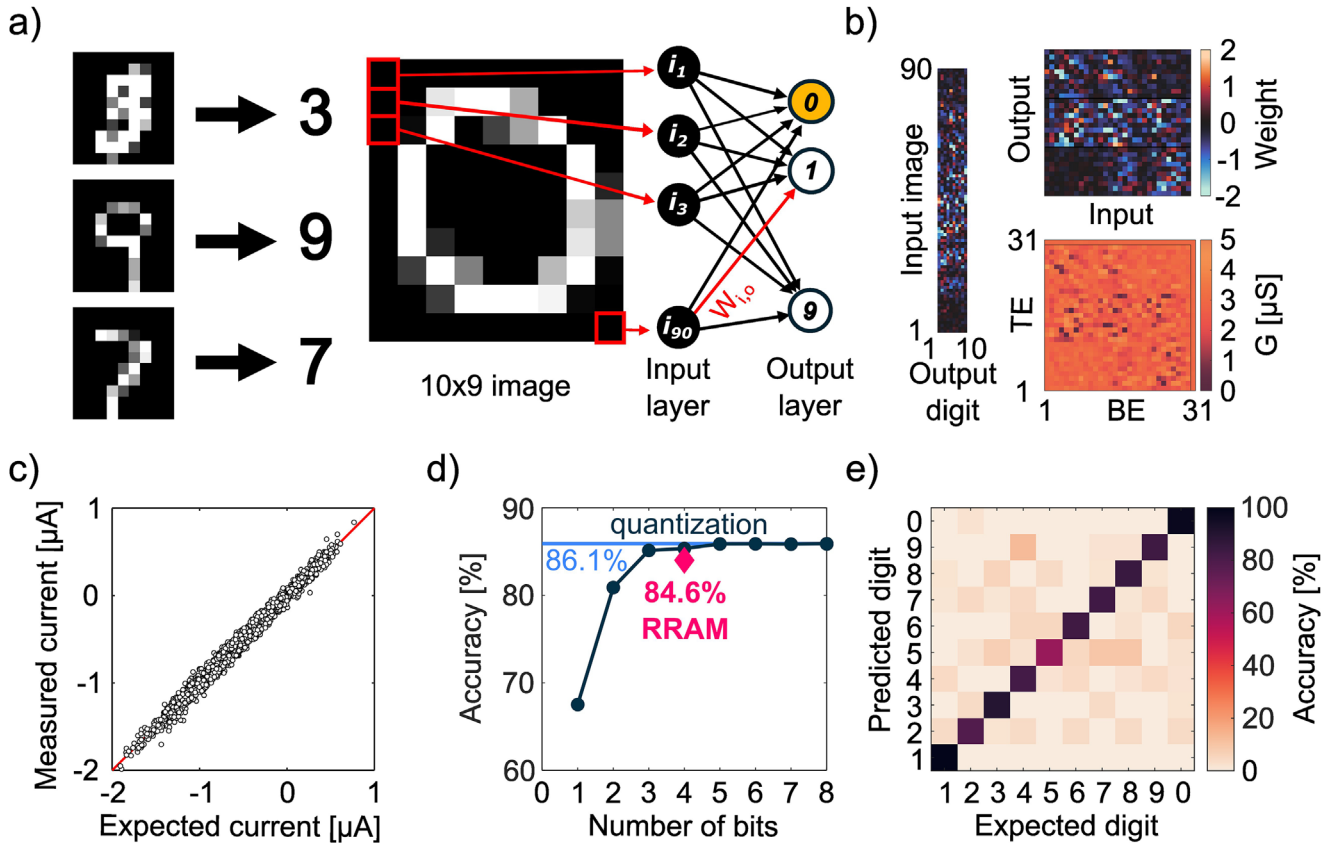


FIGURE 4 | Experimental results for pattern classification using a classifier (perceptron). (a) Sketch of the MNIST image classification task: handwritten gray-scale digits (encoded in 256 levels) are fed to a single-layer neural network (perceptron) to be recognized. Only the corresponding neuron is activated after proper training of the weights (in yellow). Input and output nodes are linked by the weighting matrix $\mathbf{W}$. (b) Weighting matrix for 10×9 image size (on the left), with 90 inputs (corresponding to the number of inputs) and 10 outputs (the number of patterns to be classified). Weights are offline trained, most of which are zeros (in black). The MNIST dataset is resized from 28×28 pixels to maximize the number of weights that can be programmed inside the 32×32 crossbar. The weighting matrix is finally partitioned into three blocks and arranged inside the matrix (on top). The total result is computed by combining the results of three MVM operations, by selecting the proper outputs. On the bottom, the programmed weight in the $0-5 \mu\text{S}$ range and $\pm 100 \text{ nS}$ of maximum error. Reference rows/columns are adopted to implement positive and negative conductance levels, thus one additional row and column are added. (c) Correlation plot between the expected currents and the measured currents during the MVM operations. Instead of using a standard activation function at the output layer, the maximum currents are selected as a response to the digit recognition. (d) Classifier accuracy as a function of the number of conductance levels (i.e., the equivalent bit-precision) and the RRAM-based accelerator performance, equivalent to a 4-bit quantized weighting matrix. (e) Confusion matrix. The highest errors for digit 5, confused with the 7 and the 8, and digit 9, confused with 4.

noise level gradually decreases according to a cooling scheme as the computation approaches the solution.

The combinatorial problem can be solved by IMC by first defining the optimization cost (or energy) function [55, 56]:

$$E = \mathbf{x}^T \mathbf{Q} \mathbf{x}$$

where $\mathbf{x}$ is a vector of binary variables and $\mathbf{Q}$ is a square Lagrangian matrix of coefficients. The coefficients of matrix $\mathbf{Q}$ depend on the specific problem, and are given by the weights between nodes in the HNN, as well as by suitable constraints in the solution, e.g., a node cannot be present in multiple solutions.

Figure 5b illustrates the quadratic assignment problem (QAP), which consists of assigning a set of objects to a set of locations in a way that minimizes the total cost of the assignment, depending on the distance between the objects and the locations

and specific flows of materials. QAP provides the backbone for airport gate assignments [57], image segmentation in processing tools, and wiring placement in PCB routing [58]. In our example, we considered a 3-facility problem which is associated with the matrix presented in Figure 5c. Positive and negative coefficients were mapped in the crossbar array by a differential conductance approach. The solution was found by iteratively applying solutions obtained by the gradient descent method, to ensure convergence, and adding noise to let the system escape local minima. Both digital noise in the input and analog RRAM noise were injected in our experiment. Figure 5d shows the evolution of the average energy function of the network as a function of the number of iterations during annealing over 100 iteration steps, together with the corresponding solution. During the evolution, the system explores solutions with different energy levels thanks to the noise effect. By cooling the noise power with the time evolution, the system is allowed to converge to a local or global minimum. Due to the stochastic nature of this approach, the system converges to the global

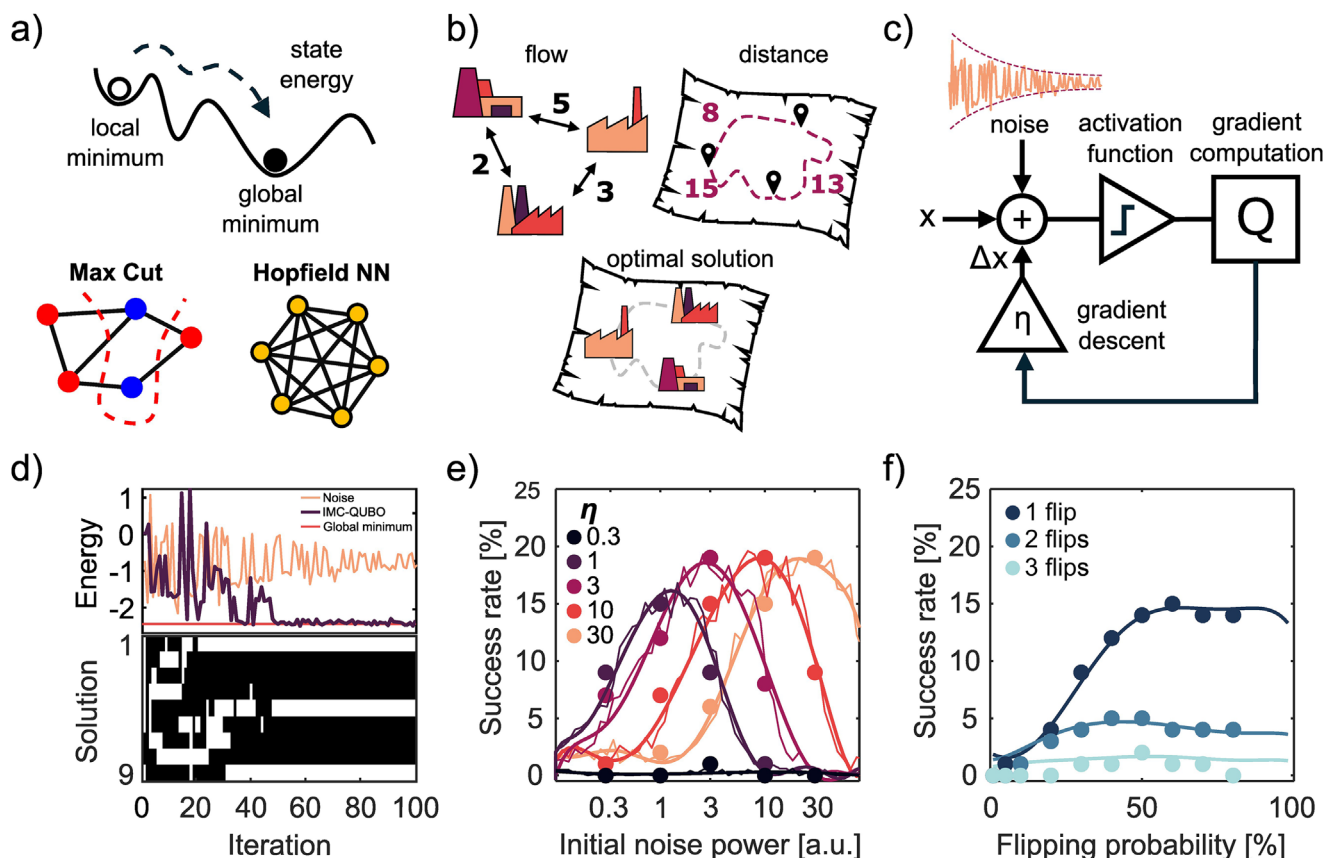


FIGURE 5 | Quadratic unconstrained binary optimization (QUBO) solver and quadratic assignment problem (QAP) accelerator based on RRAMs. (a) QUBO problems rely on the minimization of a cost function (or energy function) by searching for the global minimum of a corresponding energy landscape. This formulation is particularly useful to solve Max-Cut problems and is closely related to Hopfield neural networks. (b) QAP example: three factories must be located in three different locations, characterized by certain distances and certain flows of material existing between the locations. (c) Concept scheme of quantum-inspired, simulated annealing-based IMC-QUBO solver. At each iteration, the current circuit state is mixed with external noise and a η -weighted step along the gradient descent direction. Noise is progressively decreased during the circuit run to ensure convergence to a stable state. (d) Equivalent energy of the solution as a function of the iteration (top), and corresponding state (bottom). White and black pixels encode 1 and 0, respectively. Quantum annealing with a controlled noise model is used to run the accelerator, which tries to minimize the energy by the convergence of the solution toward the true one. (e) Comparison between simulated and measured success rate for different gradient descent rates and initial noise power to find the best operative condition. The highest success rate is reached when the two parameters are similar. (f) Success rate as a function of the flipping rate, for noise injection by random bit-flipping. The flipping probability is reduced over time to implement a cooling policy.

minimum, representing the correct solution to the problem, with a certain probability [59]. For this reason, the algorithm is applied multiple times to explore possible solutions, which are finally tested to choose the optimal one. Given a success rate S to converge to the optimal solution, the number of attempts to have a probability P that the solution is explored in those attempts is $N_t = \log(1 - P) / \log(1 - S)$. By simply adopting the gradient descent, the system remains stuck after a few iterations in a local minimum, as shown in Figure S18. Similarly, the system continuously escapes from the minima without noise cooling. The best operative condition was studied by simulations aimed at understanding the impact of various parameters, as reported in Figure S19. Figure 5e and Figure S20 show the measured and calculated success rate as a function of the initial noise power for increasing learning factor k for the gradient-descent. The success rate is maximized when the two parameters are similar, resulting in a number of attempts around $N_t = 21$.

Random flipping was studied as a method to escape local minima and reach the global minimum. To allow for system stability, the flipping probability was attenuated in time, as shown in Figure S21. As a result, the global minimum solution can be reached but with a lower probability, i.e., 15% maximum probability instead of 21%. Also, as the number of bits that can flip increases, the success rate decreases, as shown in Figure 5f.

To assess the advantage with respect to a classical CPU-based annealer, we considered optimized circuit peripherals at 14 nm technology, including 8-bit ADCs [60] for the additive Gaussian-noise implementation, 2-bit ADCs [60] for the bit-flipping implementation, and transimpedance amplifiers (TIAs) with suitably scaled feedback resistance [61]. For the CPU processor, we refer to [62] for energy-per-instruction values and suitably scale them to the same 14 nm technology used for the analog system, corresponding to an instruction energy and latency of 6 pJ and 0.2 ns, respectively, under a 5 GHz clock. We assume

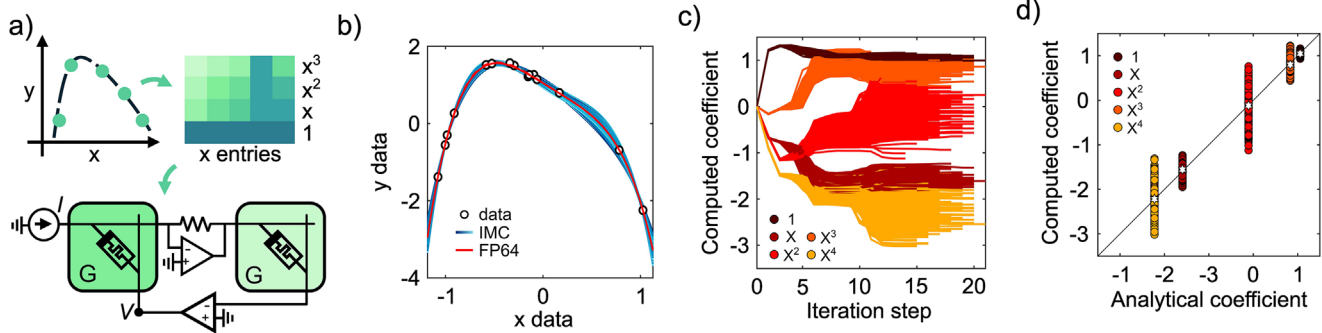


FIGURE 6 | Non-linear regression accelerator (a) Sketch of the equivalent circuit implementation to perform non-linear regression. Two matrices encode the different powers of the independent variable x while the currents map the dependent variable y . The system spontaneously converges to a set of voltages related to the coefficients of the polynomial form that best fit the data set. (b) Example of 4th-order polynomial regressions, for various IMC-based attempts (blue) and FP64 solution (red). (c) Coefficient evolution at each iteration step for different regression attempts. The error in the conductance tuning is responsible for the spread in the bundle of curves, which increases with the number of iterations, and thus with the number of MVMs. (d) Correlation plot of the coefficient values and the analytical solution. The star markers highlight the median value, which is well correlated. In the insight, the minimum number of attempts to have $\leq 1\%$ (or $\leq 10\%$) error between the median and the analytical solutions.

an $\mathcal{O}(N^2)$ computational complexity for the digital processor and normalize different implementations for a 99% solution probability. We obtain average per-MVM latencies of 257 ns, 4 ns, and 32.4 ns, and energy consumptions of 111 pJ, 1.4 pJ, and 488 pJ for the additive Gaussian noise-IMC, bit-flipping IMC, and CPU-based implementations, respectively. Overall, the IMC-based implementations enable up to $350 \times$ energy consumption and up to $8 \times$ latency improvement, further benefitting from system size upscaling.

3.4 | Non-Linear Regression

Regression is a statistical tool that can capture relationships in experimental data by retrieving the coefficients that best approximate the data trend. Regression finds particular interest in many real-life applications, such as supply and demand problems (linear regression), weight distribution in structural analysis (quadratic), chemical reaction with enzyme kinetics (cubic), or modeling certain membranes in music (quartic) [63–66]. Given a set of data $\mathbf{x}$ and $\mathbf{y}$, the general polynomial regression model can be expressed as:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \dots \quad (2)$$

where x and y are the independent and dependent variables, respectively, and β_i are the regression coefficients. The latter can be calculated by using the pseudoinverse method, namely:

$$\beta = \mathbf{X}^+ \mathbf{y} \quad (3)$$

where $\mathbf{X} = [\mathbf{1} \ \mathbf{x} \ \mathbf{x}^2 \ \dots]$. Figure 6a shows an IMC circuit for the accelerated solution of Equation (3), consisting of two identical cross-point arrays storing matrix $\mathbf{X}$ combined in a negative feedback loop. Reference rows and columns were added in the crosspoint array to implement positive and negative weights.

Figure 6b shows the measured y as a function of x for a 4th-order regression, comparing the input data mapped in the crosspoint array and the curve obtained either by the IMC regression or by

a numerical solution with FP64 precision. The figure shows the IMC solutions from 1000 attempts as well as the median of all the solutions, highlighting good agreement with the numerical solution. Figure 6c shows the computed coefficients as a function of the number of iterations, indicating a convergence in less than 20 iterations. Further results are reported in Figure S22 for the cases of linear, quadratic, cubic, and quartic regression. The computed coefficients show that the statistical spread increases at each iteration, as a consequence of the accumulation of the programming error with respect to the target. Figure 6d shows a correlation plot of the computed coefficients as a function of the ideal values, indicating that the median of the statistical distribution is in good agreement with the analytical regression. We estimated that 40 attempts are needed to achieve a 99% confidence in the median value, while 20 attempts lead to a 90% confidence. Figure S23b shows the results of IMC regression for the same problem, but with an increasing amount of noise in the data set, from 5% to 60%. As data noise increases, the error between the IMC regression and the FP64 solution increases. The median curve shows good agreement with the analytical solution in Figure S24.

4 | Conclusions

In conclusion, we presented a 32×32 crossbar array of Pt/HfO₂/Ti RRAM, demonstrating a 99.5% functionality in both initialization and set/reset processes. We showed reconfigurable weight mapping in the crossbar array with high-precision multilevel operation with an error below 3%.

To support our RRAM technology and multilevel operation methodology, we investigated four different MVM-based IMC scenarios with varying complexity, where weight matrices were encoded in memory arrays using analog-grade RRAMs. First, we demonstrate in-memory DCT for image compression and reconstruction, achieving a fidelity above 93%. Second, we implement a single-layer neural network, achieving an 84.6% classification accuracy comparable with the FP64 86.1% baseline. Third, we demonstrate a quantum-inspired QUBO solver with IMC-based

gradient descent realized by in-memory MVM, supporting the robustness of our crossbar arrays. Finally, we stress-test the programming accuracy by implementing in-memory nonlinear regression. Overall, these results corroborate the RRAM technology in terms of uniformity, multilevel operation and reliability, to support energy-efficient IMC in various real-life scenarios.

5 | Methods

5.1 | Devices Fabrication

The devices presented in this work were fabricated with CMOS-compatible processes for back-end-of-line (BEOL) integration, using standard optical lithography, metal deposition, and lift-off processes. The bottom electrode platinum film was deposited on top of CVD-grown SiO₂ through electron-beam evaporation (Evatek BAK640) after patterning AZ5214E photoresist for negative processes with a 365 nm UV light Heidelberg MLA100 maskless aligner. Then, a 70 nm SiO₂ spacer layer is grown through CVD to insulate the BEs. A second lithographic step was carried out to pattern 1 μ m diameter circular features. The pattern was used to realize vias in the spacer oxide layer by reactive ion etching in a CHF₃-O₂ plasma until the surface of the platinum film was exposed. The etching step allows to obtain cylindrical via holes that will host the memory device. A final lithographic step was performed, and a 5 nm hafnium oxide (HfO₂) layer was sputtered in Leybold LH Z400 in reactive oxygen conditions, with 12:1 Ar:O₂ ratio, 50 W with a rate of about 2 nm/min. Finally, the 130 nm Ti TE was sputtered without breaking vacuum to preserve the oxide/metal interface from unwanted oxidation and moisture. The TE was capped with 20 nm of gold to improve the pad surface finishing for bonding.

5.2 | Electrical Characterization and Experiments

Wedge bonding of RRAM arrays on custom chip carriers was performed in-house by using Bondtech 56i. A custom board with discrete components (analog switches, ADC, DAC, OpAmp, and MCU) was developed to address and properly bias the arrays and the elements, as well as for programming. For the latter, a $V/2$ scheme was used, whereby all TE and BE lines pertaining to unselected devices are grounded. For the selected device, the TE and BE lines are instead biased to $+V/2$ and $-V/2$, respectively, where V is the programming voltage. Conversely, during read operations, a read voltage V_{read} is applied to the TE of the desired cell, while all remaining TEs are grounded. The BE of the selected cell is connected to the on-board transimpedance amplifier (TIA), providing a virtual ground, whereas BEs of unselected devices are physically grounded. An on-board current compliance protection circuit was designed for the set operation. Keysight B2902A Precision Source/Measure Unit was used to calibrate the board and to carry out part of the device characterization.

5.3 | Reference Scheme for Weight Sign Representation

Differential conductance mapping usually requires twice the area to properly store the positive and negative components of

the weight. In such a scenario, considering a 32×32 array size, the maximum problem to be mapped in the array would be limited to 16×32 . To overcome this area limitation, we adopted a reference topology where one additional row and one additional column are added to the array, storing a reference conductance acting as “0” level. The reference topology consists of linearly encoding the conductances by a proper scale factor and a bias, which corresponds to a rigid shift, according to the equation:

$$\mathbf{G} = c_1 \mathbf{W} + c_2 \mathbf{J} \quad (4)$$

where $\mathbf{J}$ is the matrix of all ones, and coefficients c_1, c_2 are given by:

$$c_1 = \frac{G_{\max} - G_{\min}}{\max(\mathbf{W}) - \min(\mathbf{W})} \quad (5)$$

$$c_2 = G_{\min} - c_1 \min(\mathbf{W}) \quad (6)$$

An additional BE and TE are required to encode the rigid shift. For a 32×32 array, this results in a maximum problem size of 31×31 . Note that the shift in the mapping requires subtracting a pattern-dependent term proportional to the input voltages and the bias shift during the MVM operations. Thus, two MVMs are required to compute the right set of currents. The current voltage conversion is carried out using the c_1 factor.

5.4 | Weight Update for Single-Layer Perceptron

The single-layer neural network (perceptron) implemented in this work relies on the gradient descent method for weight update [27]. The layer was trained offline, and the resulting values are encoded in the passive resistive array. In the forward step, MVMs are performed between the input $\mathbf{x}$ and the weighting matrix $\mathbf{W}$. The output $\mathbf{z}$ is fed to the activation function, and the expected result $\mathbf{t}$ is compared to the predicted one $\mathbf{y}$, following:

$$\mathbf{z} = \mathbf{W}\mathbf{x} \text{ with } z_i = \sum_{j=1}^{90} W_{ij}x_j \quad (7)$$

$$\mathbf{y} = \sigma(\mathbf{z}) \quad (8)$$

$$\varepsilon = \|\mathbf{t} - \mathbf{y}\|_2^2 \quad (9)$$

where $\sigma(\cdot)$ is the sigmoid function. In the backward step, the weight updates $\Delta\mathbf{W}$ are performed, following the backpropagation algorithm:

$$\frac{\partial \varepsilon}{\partial y_i} = -\frac{1}{2}(t_i - y_i) \quad (10)$$

$$\delta_i = \frac{\partial \varepsilon}{\partial z_i} = \frac{\partial \varepsilon}{\partial y_i} \frac{\partial y_i}{\partial z_i} = -\frac{1}{2}(t_i - y_i) \frac{\partial \sigma(z_i)}{\partial z_i} \quad (11)$$

ALGORITHM 1 | Simulated annealing algorithm for quantum-inspired QUBO solver

Require: $\mathbf{Q} \in \mathbb{R}^{n \times n}$, N_{it} , η , p_n

```

1:      Initialize  $\mathbf{x}^{(1)} \sim \mathcal{U}(\{0, 1\}^n)$ 
2:      for  $i = 2$  to  $N_{it}$  do
3:           $\mathbf{s} \leftarrow -2\mathbf{Q}\mathbf{x}^{(i-1)}$  ▷ In-memory energy gradient MVM
4:           $\mathbf{x}^{(i)} \leftarrow \mathbf{x}^{(i-1)} - \eta\mathbf{s}$ 
5:           $\sigma \leftarrow 2p_n \exp\left(-\frac{2i}{N}\right)$ 
6:           $\mathbf{x}^{(i)} \leftarrow \mathbf{x}^{(i)} + \sigma\xi$ , where  $\xi \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
7:           $\mathbf{x}^{(i)} \leftarrow \lfloor \mathbf{x}^{(i)} \geq 0 \rfloor$ 
8:           $E^{(i)} \leftarrow (\mathbf{x}^{(i)})^\top \mathbf{Q}\mathbf{x}^{(i)}$ 
9:      end for

```

$$\Delta W_{ij} = -\eta \frac{\partial \varepsilon}{\partial W_{ij}} = -\eta \frac{\partial \varepsilon}{\partial z_i} \frac{\partial z_i}{\partial W_{ij}} = -\eta \delta_i x_j \quad (12)$$

For the sigmoid function, the first derivative can be derived as:

$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)[1 - \sigma(x)] \quad (13)$$

thus yielding the final weight update:

$$\Delta W_{ij} = \frac{\eta}{2} (t_i - y_i) \sigma(z_i) [1 - \sigma(z_i)] x_j \quad (14)$$

5.5 | Simulated Annealing Algorithm for Quantum-Inspired QUBO Solver

This section details the simulated annealing algorithm employed in simulations and measurements of the quantum-inspired QUBO solver. Let $\mathbf{Q}$ be the matrix associated with the energy function to be minimized, η the gradient descent rate, p_n the initial noise power, and N_{it} the number of iterations (See Algorithm 1).

Author Contributions

S.R., P.M., and D.I. conceived the work direction and experiments. S.R. set up the testing protocols and performed the device characterization and experiments. S.R. and D.B. fabricated the devices. P.M., M.P., F.C., and M.F. designed the algorithms. S.R., P.M., M.P. and D.I. wrote the manuscript. All authors discussed the results and reviewed the manuscript.

Acknowledgments

The authors would like to thank Claudio Somaschini, Marco Asa, Andrea Scaccabarozzi, Chiara Nava, Elisa Sogne, Stefano Bigoni, Stefano Fasoli, and Gianluca Cannetti for help in the fabrication process. This work was partially performed in Polifab, the micro- and nano-fabrication facility of Politecnico di Milano. This article has received funding from the European Research Council (ERC) under the European Union's Horizon Europe Research and Innovation Programme (Grant 101054098).

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

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support 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. Zhang, B. Gao, J. Tang, et al., "Neuro-Inspired Computing Chips," *Nature Electronics* 3, no. 7 (2020): 371–382.
3. P. Mannocci, C. Zucchelli, I. Andreoli, et al., "A Fully Integrated Analogue Closed-loop In-memory Computing Accelerator Based On Static Random-access Memory," *Nature Electronics* 9, no. 1 (2026): 200–212.
4. M. F. Ali, A. Jaiswal, and K. Roy, "In-Memory Low-Cost Bit-Serial Addition Using Commodity DRAM Technology," *IEEE Transactions on Circuits and Systems I: Regular Papers* 67, no. 1 (2020): 155–165.
5. R. Khaddam-Aljameh, M. Stanisavljevic, J. Fornit Mas, et al., "Hermes-Core—A 1.59-TOPS/mm² PCM on 14-nm CMOS In-Memory Compute Core Using 300-ps/LSB Linearized CCO-Based ADCs," *IEEE Journal of Solid-State Circuits* 57, no. 4 (2022): 1027–1038.
6. F. Carletti, M. Farronato, G. Y. C. Hu, et al., "3-D Crosspoint (3DXP) Memory Arrays with Subthreshold Operation for Low-Energy, High-Accuracy Neural Network Accelerators," *IEEE Transactions on Electron Devices* 73, no. 1 (2026): 264–270.
7. F. Carletti, M. Farronato, N. Lepri, et al., "Low-Energy, High-Accuracy Convolutional Network Inference in 3D Crosspoint (3DXP) Arrays," in *2024 IEEE European Solid-State Electronics Research Conference (ESSERC)* (2024), 412–415.
8. S. Jung, H. Lee, S. Myung, et al., "A Crossbar Array of Magnetoresistive Memory Devices for In-Memory Computing," *Nature* 601 (2022): 211–216.
9. F. Carletti, M. Y. Song, E. Ambrosi, et al., "High-Accuracy, High-Performance In-Memory Computing with High-Resistance Spin-Orbit Torque (SOT) Magnetic Memory," *IEEE Transactions on Electron Devices* 73, no. 1 (2026): 325–331.
10. M. Porzani, S. Ricci, M. Farronato, and D. Ielmini, "Programming Characteristics of Electrochemical Random Access Memory (ECRAM)—Part I: Experimental Study," *IEEE Transactions on Electron Devices* 71, no. 5 (2024): 3240–3245.
11. M. Porzani, F. Carletti, S. Ricci, M. Farronato, and D. Ielmini, "Programming Characteristics of Electrochemical Random Access Memory (ECRAM)—Part II: Physics-Based Modeling," *IEEE Transactions on Electron Devices* 71, no. 5 (2024): 3246–3251.
12. J. Cui, F. An, J. Qian, et al., "CMOS-Compatible Electrochemical Synaptic Transistor Arrays for Deep Learning Accelerators," *Nature Electronics* 6, no. 4 (2023): 292–300.
13. K. Noh, H. Kwak, J. Son, et al., "Retention-Aware Zero-Shifting Technique for Tiki-Taka Algorithm-Based Analog Deep Learning Accelerator," *Science Advances* 10, no. 24 (2024): ead13350.
14. T. Wang, J. Meng, X. Zhou, et al., "Reconfigurable Neuromorphic Memristor Network for Ultralow-Power Smart Textile Electronics," *Nature Communications* 13, no. 1, (2022): 7432.
15. G. Milano, G. Pedretti, K. Montano, et al., "In Materia Reservoir Computing with a Fully Memristive Architecture Based on Self-Organizing Nanowire Networks," *Nature Materials* 21, no. 2 (2021): 195–202.
16. A. Milozzi, S. Ricci, and D. Ielmini, "Memristive Tonotopic Mapping with Volatile Resistive Switching Memory Devices," *Nature Communications* 15, no. 1 (2024): 2812.

17. 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.
18. S. Choi, J. H. Shin, J. Lee, P. Sheridan, and W. D. Lu, "Experimental Demonstration of Feature Extraction and Dimensionality Reduction Using Memristor Networks," *Nano Letters* 17, no. 5 (2017): 3113–3118.
19. H. Li, S. Wang, X. Zhang, et al., "Memristive Crossbar Arrays for Storage and Computing Applications," *Advanced Intelligent Systems* 3, no. 9 (2021): 2100017.
20. Y. R. Kim, D. H. Shin, N. Ghenzi, et al., "Fully Resistive In-Memory Cryptographic Engine with Dual-Polarity Memristive Crossbar Array," *Advanced Functional Materials* (2024).
21. K.-H. Kim, S. Gaba, D. Wheeler, et al., "A Functional Hybrid Memristor Crossbar-Array/CMOS System for Data Storage and Neuromorphic Applications," *Nano Letters* 12, no. 1 (2011): 389–395.
22. J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive Devices for Computing," *Nature Nanotechnology* 8, no. 1 (2012): 13–24.
23. S. Kumar, X. Wang, J. P. Strachan, Y. Yang, and W. D. Lu, "Dynamical Memristors for Higher-Complexity Neuromorphic Computing," *Nature Reviews Materials* 7, no. 7 (2022): 575–591.
24. F. Cai, J. M. Correll, S. H. Lee, et al., "A Fully Integrated Reprogrammable Memristor–CMOS System for Efficient Multiply–Accumulate Operations," *Nature Electronics* 2, no. 7 (2019): 290–299.
25. 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* 521, no. 7550 (2015): 61–64.
26. D. Bridaroli, C. Zucchelli, P. Mannoce, et al., "3-D Vertical Resistive Switching Random Access Memory (3D-VRRAM) with Multilevel Programming for High-Density, Energy-Efficient In-Memory Computing," *IEEE Transactions on Electron Devices* 72, no. 5 (2025): 2677–2684.
27. H. Kim, M. R. Mahmoodi, H. Nili, and D. B. Strukov, "4K-Memristor Analog-Grade Passive Crossbar Circuit," *Nature Communications* 12, no. 1 (2021): 5198.
28. S. Ricci, P. Mannoce, M. Farronato, S. Hashemkhani, and D. Ielmini, "Forming-Free Resistive Switching Memory Crosspoint Arrays for In-Memory Machine Learning," *Advanced Intelligent Systems* 4, no. 8 (2022): 2200053.
29. H. Jeong, S. Han, S.-O. Park, et al., "Self-Supervised Video Processing with Self-Calibration on an Analogue Computing Platform Based on a Selector-Less Memristor Array," *Nature Electronics* (2025).
30. W. Sun, B. Gao, M. Chi, et al., "Understanding Memristive Switching via In Situ Characterization and Device Modeling," *Nature Communications* 10, no. 1 (2019): 3453.
31. H. Kang, J. Park, D. Lee, et al., "Two- and Three-Terminal HfO₂-Based Multilevel Resistive Memories for Neuromorphic Analog Synaptic Elements," *Neuromorphic Computing and Engineering* 1, no. 2 (2021): 021001.
32. S. Chen and I. Valov, "Design of Materials Configuration for Optimizing Redox-Based Resistive Switching Memories," *Advanced Materials* 34, no. 3 (2021): 2105022.
33. H. Yeon, P. Lin, C. Choi, et al., "Alloying Conducting Channels for Reliable Neuromorphic Computing," *Nature Nanotechnology* 15, no. 7 (2020): 574–579.
34. S. Kim, K. Hong, H. Kim, M. Kim, and W. Y. Choi, "Overshoot-Suppressed Memristor Array with AlN Oxygen Barrier for Low-Power Operation in the Intelligent Neuromorphic Systems," *Advanced Intelligent Systems* 6, no. 8 (2024): 2300797.
35. F. Zahoor, T. Z. Azni Zulkifli, and F. A. Khanday, "Resistive Random Access Memory (RRAM): an Overview of Materials, Switching Mechanism, Performance, Multilevel Cell (MLC) Storage, Modeling, and Applications," *Nanoscale Research Letters* 15, no. 1 (2020): 90.
36. P. Mannoce, M. Farronato, N. Lepri, et al., "In-Memory Computing with Emerging Memory Devices: Status and Outlook," *APL Machine Learning* 1, no. 1 (2023): 010902.
37. A. James and B. Choubey, *Selected Topics in Intelligent Chips with Emerging Devices, Circuits and Systems*, 1 ed. (River Publishers, New York, 2023).
38. N. R. Shanbhag and S. K. Roy, "Benchmarking In-Memory Computing Architectures," *IEEE Open Journal of the Solid-State Circuits Society* 2 (2022): 288–300.
39. "International Roadmap for Devices and Systems (IRDS) 2024," (2024), <https://irds.ieee.org/>.
40. N. Lepri, M. Baldo, P. Mannoce, 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.
41. N. Lepri, A. Glukhov, and D. Ielmini, in *2022 IEEE International Reliability Physics Symposium (IRPS)* (IEEE, Dallas, TX, USA), ISBN 978-1-6654-7950-9, 2022 3C.2-1-3C.2-6, <https://ieeexplore.ieee.org/document/9764486/>.
42. N. Lepri, A. Glukhov, P. Mannoce, M. Porzani, and D. Ielmini, "Compact Modeling and Mitigation of Parasitics in Crosspoint Accelerators of Neural Networks," *IEEE Transactions on Electron Devices* 71, no. 3 (2024): 1900–1906.
43. P. Mannoce, G. Larelli, M. Bonomi, and D. Ielmini, "Achieving High Precision in Analog In-Memory Computing Systems," *npj Unconventional Computing* 3, no. 1 (2026): 1.
44. M. Le Gallo, R. Khaddam-Aljameh, M. Stanisavljevic, et al., "A 64-Core Mixed-Signal In-Memory Compute Chip Based on Phase-Change Memory for Deep Neural Network Inference," *Nature Electronics* 6, no. 9 (2023): 680–693.
45. 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.
46. M. Dazzi, A. Sebastian, T. Parnell, P. A. Francese, L. Benini, and E. Eleftheriou, "Efficient Pipelined Execution of CNNs Based on In-Memory Computing and Graph Homomorphism Verification," *IEEE Transactions on Computers* 70, no. 6 (2021): 922.
47. H. Wen, L. Ma, L. Liu, et al., "High-Quality Restoration Image Encryption Using DCT Frequency-Domain Compression Coding and Chaos," *Scientific Reports* 12, no. 1 (2022): 16523.
48. D. Salomon, ed., *Data Compression: The Complete Reference*, 4th ed. (Springer London, 2007).
49. Y. Feng, Y. Zhang, Z. Zhou, et al., "Memristor-Based Storage System with Convolutional Autoencoder-Based Image Compression Network," *Nature Communications* 15, no. 1 (2024): 1132.
50. P. Yao, H. Wu, B. Gao, et al., "Fully Hardware-Implemented Memristor Convolutional Neural Network," *Nature* 577, no. 7792 (2020): 641–646.
51. 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.
52. Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature* 521, no. 7553 (2015): 436–444.
53. P. Yao, H. Wu, B. Gao, et al., "Face Classification Using Electronic Synapses," *Nature Communications* 8, no. 1 (2017): 15199.
54. F. Cai, S. Kumar, T. Van Vaerenbergh, et al., "Power-Efficient Combinatorial Optimization Using Intrinsic Noise in Memristor Hopfield Neural Networks," *Nature Electronics* 3, no. 7 (2020): 409–418.
55. F. Glover, G. Kochenberger, R. Hennig, and Y. Du, "Quantum Bridge Analytics I: A Tutorial on Formulating and Using QUBO Models," *Annals of Operations Research* 314, no. 1 (2022): 141–183.

56. M. Jiang, K. Shan, C. He, and C. Li, "Efficient Combinatorial Optimization by Quantum-Inspired Parallel Annealing in Analogue Memristor Crossbar" *Nature Communications* 14, no. 1 (2023): 5927.
57. M. E. Narciso and M. A. Piera, "Robust Gate Assignment Procedures from an Airport Management Perspective," *Omega* 50 (2015): 82–95.
58. R. Zamani and M. Amirghasemi, *A Self-adaptive Nature-Inspired Procedure for Solving the Quadratic Assignment Problem* (Springer Singapore, 2020), 119–147.
59. Z. Fahimi, M. R. Mahmoodi, H. Nili, V. Polishchuk, and D. B. Strukov, "Combinatorial Optimization by Weight Annealing in Memristive Hopfield Networks," *Scientific Reports* 11, no. 1 (2021): 16383.
60. M. J. Marinella, S. Agarwal, A. Hsia, et al., *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 8, no. 1 (2018): 86–101.
61. B. Feinberg, R. Wong, T. P. Xiao, et al., in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (IEEE, Seoul, Korea (South), 2021), 761–774, ISBN 978-1-6654-2235-2, <https://ieeexplore.ieee.org/document/9407108/>.
62. M. Horowitz, in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)* (IEEE, San Francisco, CA, USA, 2014), 10–14, ISBN 978-1-4799-0920-9 978-1-4799-0918-6 <http://ieeexplore.ieee.org/document/6757323/>.
63. D. M. Bates and D. G. Watts, *Nonlinear Regression Analysis and Its Applications* (Wiley, 1988).
64. X. Xiao, E. P. White, M. B. Hooten, and S. L. Durham, "On the Use of Log-Transformation vs. Nonlinear Regression for Analyzing Biological Power Laws," *Ecology* 92, no. 10 (2011): 1887–1894.
65. S. V. Archontoulis and F. E. Miguez, "Nonlinear Regression Models and Applications in Agricultural Research," *Agronomy Journal* 107, no. 2 (2015): 786–798.
66. J. Yang, R. Mao, M. Jiang, et al., "Efficient Nonlinear Function Approximation in Analog Resistive Crossbars for Recurrent Neural Networks," *Nature Communications* 16, no. 1 (2025): 1136.

Supporting Information

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

Supporting File: aelm70500-sup-0001-SuppMat.pdf.