Nature Communications

https://doi.org/10.1038/s41467-026-69958-0

Article in Press

A hardware-adaptive learning algorithm for superlinear-capacity associative memory on memristor crossbars

Received: 8 May 2025

Accepted: 14 February 2026

Cite this article as: He, C., Jiang, M., Shan, K. *et al.* A hardware-adaptive learning algorithm for superlinear-capacity associative memory on memristor crossbars. *Nat Commun* (2026). https://doi.org/10.1038/s41467-026-69958-0

Chengping He, Mingrui Jiang, Keyi Shan, Szu-Hao Yang, Zefan Li, Shengbo Wang, Giacomo Pedretti, Jim Ignowski & Can Li

A Hardware-Adaptive Learning Algorithm for Superlinear-Capacity Associative Memory on Memristor Crossbars

Chengping He[1,2], Mingrui Jiang[1,2], Keyi Shan[1,2], Szu-Hao Yang[1,2], Zefan Li[1,2], Shengbo Wang[1,2], Giacomo Pedretti[3], Jim Ignowski[3], and Can Li[1,2,*]

[1]Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong SAR, China
[2]Center for Advanced Semiconductors and Integrated Circuits, The University of Hong Kong, Hong Kong SAR, China
[3]Hewlett Packard Labs, Hewlett Packard Enterprise, Milpitas, CA, USA
[*]Email: canl@hku.hk

ABSTRACT

The human brain recalls complete patterns from partial cues via associative memory, but Hopfield neural networks emulating this process are inefficient on conventional hardware, and prior memristor-based implementations are vulnerable to device defects and have limited capacity, particularly for continuous patterns. We introduce a hardware-adaptive learning algorithm that incorporates experimentally calibrated device constraints during training and validate it on an integrated memristor crossbar compute-in-memory platform. The approach improves defect tolerance and effective capacity, achieving threefold higher capacity than a pseudo-inverse baseline at 50% stuck-at faults. The same framework extends to scalable multilayer architectures supporting binary and continuous-valued patterns, where we observe superlinear capacity scaling on correlated data ($\propto N^{1.49}$ and $\propto N^{1.74}$, respectively). Leveraging crossbar parallelism with synchronous updates, the implementation reduces energy by 8.8$\times$ and latency by 99.7% for 64-dimensional patterns versus asynchronous schemes. These results provide a practical algorithm–hardware co-design for robust, efficient Hopfield-style associative recall.

Introduction

The notion of drawing inspiration from the complex mechanisms of the biological brain has sparked a new era of breakthroughs in the field of artificial intelligence (AI)[1]. These advancements encompass a diverse array of innovations, spanning from computational vision (CV)[2] to natural language processing (NLP)[3]. Aside from the crucial functionalities of pattern recognition and classification, the biological brain also exhibits a remarkable ability for associative recall, i.e., retrieving a complete memory from partial or corrupted cues, as exemplified by Pavlov's dog experiment[4]. An associative memory network, also referred to as a type of recurrent neural network, retrieves a stored pattern from incomplete or corrupted inputs through iterative state evolution (pattern completion), which is conceptually different from conventional CAM-style one-shot lookup that outputs a match/mismatch signal or an address[5].

Various models have been proposed in intelligent systems to implement associative memory. One of the most significant models is the Hopfield neural network (HNN)[6–12], which constitutes a fully connected network capable of utilizing local minima to store memory patterns (Figure1 (a)). When a corrupted or partial pattern is inputted into the system, the system's state automatically decreases the energy based on the update rule. Consequently, the system reaches a stable state, representing a local minimum and the location where patterns are stored. As a result, the HNN can effectively retrieve the original patterns, making it a valuable associative memory system. Due to its energy-decreasing nature, the HNN finds applications in diverse fields, including solving optimization problems by introducing perturbations to reach the global minimum instead of local ones in associative memory[8,13]. However, implementing these energy-minimization processes efficiently remains challenging in classical computers due to their serial and digital processing nature and their separation of memory and process unit (Figure 1 (b)).

Memristors, a type of non-volatile memory, show a great potential for neuromorphic computing, including the implementation of associative memory. They offer co-located memory and computing, massive parallelism, and thus high speed and low energy consumption[14–16] (Figure 1c). Many neuromorphic computing applications have been successfully implemented using the memristor systems in the recent years, such as signal processing[17,18], deep neural networks[19–27], optimization problems[28–30], scientific computing[31,32], and hardware security[33,34]. The Hopfield neural network can also benefit significantly from the parallel computation capability inherent in memristor systems. Many important applications of the Hopfield neural network have been successfully implemented in memristor systems, particularly in optimization tasks. Apart from the demonstration of using HNN for optimization, the implementation of HNN on memristors is inherently suited to mimic important functionalities of biological systems. Previous studies have demonstrated HNN implementations in various memristor technologies, including phase change memory (PCM), resistive random-access memory (RRAM) for associative memory, and ferroelectric devices in different crossbar array sizes ranging from 3×3 to 128×8[35–41].

Despite these successes, significant challenges persist (Figure 1(d)). First, previous implementations are highly sensitive to device non-idealities, such as stuck-at-fault devices. This sensitivity arises because current offline learning algorithms for associative memory in HNNs don't account for any hardware information, such as device non-idealities, which is one of reasons preventing memristive associative memory from practical use. Second, the capacity of previous HNN models has been significantly constrained by the synaptic structure of the system. Since traditional HNNs are based on a single-layer architecture with the same number of input and output neurons, the capacity is inherently limited by the number of input neurons. For example, associative memory with Hebbian learning, the most classical learning rule, can only store up to $0.14\times$N patterns, where N is the number of neurons[6]. Therefore, once the number of input neurons is defined, the system's capacity becomes restricted, severely limiting the amount of information it can store and recall effectively. Third, the earlier learning rule imposes stringent limitations on associating binary patterns. This restricts the functionality of associative memory, since many real-world patterns are continuous values, not binary.

To overcome these limitations, we propose a hardware-adaptive learning algorithm that directly targets defect tolerance, flexibility, and continuous-mode compatibility, while leveraging the parallelism of memristor crossbars. With hardware-adaptive training, we explicitly account for device non-idealities during learning, enabling substantially improved defect tolerance: for example, on MNIST, our method maintains $3\times$ higher effective capacity than the state-of-the-art pseudoinverse baseline when 50% of devices are stuck-at faults. Our approach can also be extended to a multilayer structure, which our observations

show substantial capacity benefits for binary patterns and enables associative recall for continuous-valued patterns. For binary patterns, we empirically observe a superlinear relationship between capacity and input neuron dimension ($\propto N^{1.49}$) on correlated datasets like MNIST, while traditional single-layer HNNs show only a linear correlation ($\propto N^{1.06}$). Adding a hidden layer also increases architectural flexibility and can reduce memristor usage by up to 95% when storing the MNIST dataset. The multilayer structure further supports continuous patterns, where we empirically observe superlinear scaling ($\propto N^{1.74}$). We experimentally validate our adaptive learning algorithm on an integrated RRAM crossbar platform used as a representative compute-in-memory system, and demonstrate improved energy efficiency and lower latency compared with prior asynchronous update schemes. Specifically, our synchronous update can reduce processing time by 99.4% and improve energy efficiency by a factor of 2.68× to 2.76× compared to previous asynchronous updates. Additionally, the multilayer design offers 2.53× to 3.28× higher energy efficiency and operates 1–2× faster than single-layer implementations, marking a substantial step forward in the development of scalable, energy-efficient memristor-based associative memory technologies.

Results

Hopfield Neural Network For Associative Memory

The Hopfield Neural Network was first introduced by Hopfield in 1982[6]. It is characterized by being a recurrent fully connected neural network, with a decreasing energy function during the recurrent state updates. This energy function is typically described by the Ising form energy equation:

$$H = -\frac{1}{2}\sum_{i,j}\left(W_{ij}x_i x_j - b_i x_i\right). \tag{1}$$

Here, the x_i represents the state of neuron i, which can be either 1 and -1, indicating whether the neuron is activated or not. The terms W_{ij} and b_i are the connection weights between different neurons and the threshold, respectively. The system updates neurons states using the following formula:

$$\mathbf{x}_{t+1} = \mathrm{sgn}(\mathbf{W}\mathbf{x}_t - \mathbf{b}). \tag{2}$$

This update rule guarantees that the energy of the system will decrease after each update. The network will eventually reach a stable state, which is a local minimum of the energy function. In many standard formulations, the threshold vector is set to zero, i.e., $\mathbf{b} = \mathbf{0}$, yielding the simplified update $\mathbf{x}_{t+1} = \mathrm{sgn}(\mathbf{W}\mathbf{x}_t)$. We retain $\mathbf{b}$ for completeness, since it can model neuron thresholds or bias offsets (e.g., hardware reference levels) when needed. Unless otherwise specified, we set $\mathbf{b} = \mathbf{0}$ throughout this work. When using the Hopfield neural network for associative memory, patterns are stored in the local minima of the system. If a corrupted pattern is input, the network will evolve towards a stable state through neuron updates, retrieving the stored pattern once stability is reached.

The patterns are stored in the local minima of the system by configuring the weights of the system. The most well-known learning rule for the Hopfield neural network is a biological-inspired Hebbian learning rule[42], which is a simple correlation-

based learning rule, but this training method suffers from limited capacity problems. Many other learning rules, such as the Storkey learning rule[43] and the pseudo-inverse learning rule[44], have been proposed to address the capacity limitations of the Hebbian learning rule. However, these methods are offline algorithms that train the weights offline based solely on stored patterns, without an adaptive training process. This limits their adaptability to the non-idealities when they are implemented in memristor-based systems. Due to the inherent variability and stuck-at faults in memristors, the mapped weights can significantly deviate from those learned in software. Moreover, offline training necessitates precise weight mapping, exacerbating the impact of device variations on weight accuracy.

Hardware-Adaptive Learning Algorithm for Associative Memory

To overcome the limitations of previous learning algorithms, we propose a hardware-adaptive learning rule specifically designed for memristor-based Hopfield Neural Networks. Our approach moves beyond commonly used software-level learning baselines[42–45] which are typically defined under ideal arithmetic and do not explicitly account for device constraints or non-idealities, by focusing on practical performance and hardware realization. In particular, our adaptive training achieves measurable improvements in recall robustness and capacity on both correlated and uncorrelated pattern sets, and it remains effective when deployed on physical memristor arrays by explicitly accounting for experimental non-idealities (e.g., defects and conductance constraints). Moreover, the same formulation naturally extends to multilayer architectures, enabling empirically observed superlinear capacity scaling and continuous-valued associative recall—capabilities not exhibited by conventional single-layer formulations.

More specifically, for states not at the energy minimum , the neuron state changes after an update. However, for states at the minimum, the neuron state remains unchanged after an update, indicating that $\mathbf{x}_{t+1} = \mathrm{sgn}(\mathbf{W}\mathbf{x}_t - \mathbf{b}) = \mathbf{x}_t$. Thus, at the minimum state $\mathbf{x}_t = \mathrm{sgn}(\mathbf{W}\mathbf{x}_t - \mathbf{b})$. Therefore, each stored pattern $\mathbf{x}^{(p)}$ should be a stable fixed point of the dynamics. Accordingly, we should adjust the weight so that the system stabilizes at these stored patterns. This can be mathematically expressed as minimizing the following objective function:

$$\min_{\mathbf{W}} \sum_{p} \mathrm{Distance}\left(\mathbf{x}^{(p)}, \mathrm{sgn}\left(\mathbf{W}\mathbf{x}^{(p)} - \mathbf{b}\right)\right). \tag{3}$$

where the Distance represents the distance function used to measure the difference between the stored pattern $\mathbf{x}^{(p)}$ and the system output $\mathrm{sgn}\left(\mathbf{W}\mathbf{x}^{(p)} - \mathbf{b}\right)$, such as L2 distance or L1 distance.

Since the gradient of the sgn function is difficult to compute due to its derivative being the Dirac delta function, we reformulate the problem as follows:

$$\min_{\mathbf{W}} \sum_{p} \mathrm{Distance}\left(\mathbf{x}^{(p)}, \tanh\left(\lambda\left(\mathbf{W}\mathbf{x}^{(p)} - \mathbf{b}\right)\right)\right). \tag{4}$$

In this revised equation, because the sgn function's discontinuity inhibits gradient descent, we adopt the tanh function as a smooth approximation, an approach well-established in prior work[46,47]. The parameter λ in equation 4 controls the steepness of the objective function; for simplicity, we set $\lambda = 1$. By employing this modified loss function, we can use gradient

descent to minimize the loss, thereby ensuring that the stored patterns correspond to the minima of the system. This approach demonstrates the feasibility of training the system's weights using an objective function approach, offering a potential solution to the challenges faced by traditional learning algorithms.

Since the learning algorithms of the Hopfield Neural Network (HNN) can be translated into a loss function, it becomes feasible to employ gradient descent to minimize this function. This enables the weights to be adjusted naturally to adapt to device defects when offline trained based on a real hardware-calibrated model. The process begins with a characterization scan of the physical memristor array to record the positions of stuck-at-fault devices. Any device that fails to reach a minimum conductance threshold (e.g., 5 μS after programming) is marked as a permanent stuck-at-fault. During training, we enforce a fixed mask that sets the corresponding weights to zero. The training algorithm then automatically learns to compensate for the masked-out devices. After offline training is complete, the learned weights are programmed onto the physical crossbar to carry out associative memory tasks. Further details of the training process are provided in the Supplementary Note 1.

Another advantage of our adaptive learning algorithm is the ability to apply techniques from deep neural networks to the Hopfield Neural Network (HNN), such as using a multilayer structure. Classic HNNs are limited by their single-layer structure, where the number of neurons matches the number of input patterns, and the synaptic connections are constrained by the square of the neuron count. This setup heavily restricts the system's capacity and capability. In contrast, our learning algorithm can make use of a multilayer structure, leveraging the compression capabilities of deep neural networks and the hidden layers inherent in such a structure, and thus offers greater flexibility and the potential for improved capability and feasibility.

Memristor-based Associative Memory System

We demonstrated the hardware-adaptive learning algorithm for associative memory in our integrated memristor system, a task distinct from classification, as shown in Figure 2(a). The algorithm generates learned weights that represent stored patterns, which we map to the memristor crossbar by converting them to conductance values ranging from 0 to 150 μS. In our experiments, we input corrupted patterns into the system via the integrated digital-to-analog converters (DACs) shown in Figure 2(b). These inputs are processed by one of our 64×64 one-transistor-one-memristor (1T1M) crossbars (Figure 2(c)), which performs matrix-vector multiplication in the analog domain. The resulting outputs are read out by integrated transimpedance amplifiers (TIAs) and analog-to-digital converters (ADCs). The neuron states are then updated in a mixed-signal realization. The nonlinear activations (e.g., sign or tanh) and current update logic are implemented digitally after the ADC conversion. This approach provides flexibility to allow us to experimentally validate both binary and continuous patterns with the same hardware. However, we acknowledge that the AD/DA conversion is the primary bottleneck for system energy and latency. Future optimization can eliminate this by using fully-analog on-chip neuron circuits, such as comparators for the sign function and differential pairs for the tanh function, for a more efficient system (see Supplementary Note 2 for detailed circuit concepts). While traditional Hopfield neural networks use asynchronous updates[35,37], our memristor system implements synchronous updates (update all neurons at the same time), which better leverages the parallel computing capabilities of the memristor crossbar.

In this memristive associative memory system, the peripheral circuit and selector transistor of the integrated memristor system are designed following a 180 nm CMOS design rule, with fabrication completed by a commercial foundry. Following CMOS fabrication, we integrated the memristor devices onto the chip in-house using back-end processes. Full details on our

integration process are available in the Method section and our previous publication[48], the picture of our test chip is shown in Supplementary Figure 1. Figure 2(d) illustrates the direct-current (DC) current-voltage (I-V) characteristics of the integrated memristor across 100 Set and Reset cycles, showing minimal cycle-to-cycle variation. As demonstrated in Figure 2(e), the devices can be programmed to various analog conductance states with linear I-V relationships. These multilevel states exhibit excellent retention with no obvious systematic conductance drift, as verified by our retention test on a programmed 64×64 array over 10^5 seconds (more than one day) shown in Figure 2(f).

Pattern Retrieval Experiment with Hardware-Adaptive Learning Algorithm

We perform experiments to evaluate the performance of our memristor-based associative memory system in retrieving patterns. The stored patterns are taken from the Modified National Institute of Standards and Technology (MNIST) database, which contains 60,000 different hand-written patterns for digits 0-9. We randomly select different numbers of patterns and store them in the weight matrix using the previously described training algorithm, while accounting for device non-idealities in the system (Figure 3a). To fit the crossbar size, we preprocess the 28×28 images by cropping them to 24×24 and downsampling to 8×8 via bicubic interpolation. The images are then binarized and reshaped into 64×1 vectors, with each Hopfield network neuron representing one pixel.

To illustrate how the system works, we first store ten patterns, one for each MNIST digit. After offline training, to represent both positive and negative synaptic weights with non-negative memristor conductance, each synaptic weight is implemented as a differential pair of memristors, with the value encoded by the conductance difference of the pair. The nonlinear activation functions are implemented digitally after ADC conversion. Figure 3(b, c) shows the target conductance map and the readout target conductance map after experimentally programming them into the crossbars, and we can see that the readout conductance is very close to the target conductance despite minor variations. After configuring the memristor crossbar to store the patterns, we test pattern retrieval by inputting corrupted patterns into the system (more details of hardware implementation in Supplementary Note 2). We first generate the corrupted patterns by randomly flipping 10% of the pixels in the stored patterns, and then input these corrupted patterns into the system. The neuron states update iteratively based on the crossbar output, with experimental values closely matching expected outputs, as shown in Figure 3(d) . As the iterations progress, the system gradually reaches a stable state (neuron states do not flip anymore), which we read as the final output. Figure 3(e) shows corrupted patterns as the input and the stable state we read from the memristor system after only five iterations, which reveals no noticeable difference from the stored patterns. The retrieval process update all neurons simultaneously (synchronous updates) to fully exploit the system's parallelism. While synchronous updates can theoretically cause oscillations, our analysis shows they have a negligible impact on the final retrieval quality under practical conditions, while offering substantial improvement in latency and energy efficiency (see Supplementary Note 3 for a detailed comparison). The effectiveness of the synchronous update is further detailed in Supplementary Figure 2, where most corrupted pixels are corrected within two to three iterations. In all experiments, we run retrieval for at most $T_{\max} = 10$ iterations and use the state at iteration $T_{\max}$ as the final recalled pattern. Most corrupted pixels are successfully recovered within two or three steps. We also compute the energy and cosine similarity of the neuron states to evaluate the retrieval process quantitatively. The cosine similarity distance quantifies the difference between stored and retrieved binary patterns, which is the key metric for this associative memory task, different from the classification accuracy. A cosine similarity of closer to one indicates better associative recall. Figure 3(f) shows the energy and cosine similarity as

functions of the number of iterations for the digit '1'. As the iterations progress, the energy decreases and the cosine similarity increases, both stabilizing after five iterations. This indicates convergence to a minimum energy state and successful pattern retrieval.

One of the most important metrics for an associative memory is how many patterns can be stored, or in other words, the capacity of the system. As the number of stored patterns increases, it becomes increasingly challenging to retrieve the correct pattern because the encoded local minima become closer to each other, leading to a higher probability of convergence to the wrong state. Therefore, the learning algorithm has an impact on how patterns are distributed in the energy landscape, and thus on the capacity of the system. Our experimental results demonstrated that our adaptive learning algorithm also significantly improves the system's capacity compared to previous algorithms. We randomly sample patterns from the MNIST dataset (note that consistent trends were observed across EMNIST and random pattern datasets, see Supplementary Figures 3-5, for details) and conduct the experiment 20 times to ensure reliable comparisons. Figure 3(g) shows the cosine similarity (datapoints and solid lines showing the mean value, with shaded region representing the inter-quartile across 20 experiments between stored and retrieved patterns as a function of the number of stored patterns, where a value closer to 1 indicates better retrieval. To provide an apples-to-apples comparison, we benchmark against established SOTA associative memory algorithms that share the same task. For these baselines, including the pseudo-inverse, Hebbian, and equilibrium propagation algorithms[49,50], the cosine similarity between stored and retrieved patterns rapidly declines when the number of stored patterns exceeds a certain value. This decline leads to the 'catastrophic forgetting', where all stored patterns are effectively erased once the system's capacity is surpassed[51,52]. In contrast, our adaptive learning algorithm demonstrates much more stable performance.

We further evaluate the impact of stuck-at faults on the system's data integrity to demonstrate the superior defect tolerance of our adaptive learning algorithm. Figure 3(h) illustrates the effect of stuck-at fault devices on the retrieval quality of the final results. For simplicity, we compare our method only with the state-of-the-art (SOTA) pseudo-inverse approach, which shows good robustness to the device non-idealities.(additional comparisons with other approaches can be found in the supplementary Figure 6). As the stuck-at-fault ratio increases, the retrieved similarity degrades; however, our algorithm maintains higher performance, tolerating fault ratios up to 50% compared to the baseline's 35%.

We also analyze the impact of stuck-at faults on system capacity. To make the consistent capacity comparison across learning algorithms consistent, we define the system capacity as the maximum number of patterns that can be retrieved with a cosine similarity≥ 0.99 when the input cue is corrupted with a 5% bit-flip probability. This definition is stricter and more practically relevant than the classical definition of Hopfield network capacity, which evaluates the maximum number of stored patterns that exist as stable fixed points in a noiseless system (0% bit-flip probability). In our experiments, the classical definition yields a capacity of $0.132N$ for the Hebbian learning rule (Supplementary Note 4), consistent with the theoretical value of $0.136N$ reported in[53]. By contrast, our stricter definition probes the basin of attraction by testing retrieval from corrupted inputs which is more practical relevant. This is a more challenging test, and as expected, it yields a slightly lower capacity of $0.127N$ than this noiseless fixed-point evaluation. We use this consistent definition across all experiments in the manuscript for fair comparison. (More details are discussed in Supplementary Note 4). Figure 3(i) illustrates the scaling relationship between capacity and the number of neurons. While both our and the previously reported SOTA methods show a linear increase in capacity, our approach achieves twice the capacity of the state-of-the-art (SOTA) method. As the number of neurons increases, the capacity growth slows slightly due to higher pattern correlation in MNIST, which may lead to a

slight performance decrease. Figure 3(j) compares how storage capacity degrades with increasing stuck-at-fault ratio across different learning methods. Compared with previous approaches, our hardware-adaptive training more effectively exploits the remaining functional devices by explicitly incorporating stuck-at constraints into optimization, thereby reducing reliance on defective synapses and preserving stable attractors. As a result, capacity decreases more gradually with fault ratio, indicating a graceful-degradation behavior in which recall quality can be maintained while the number of stored patterns is reduced as hard errors (stuck-at-fault) accumulate. Notably, at a 50% stuck-at fault ratio, our method maintains three times the capacity of the SOTA (115 patterns versus 35), demonstrating substantially stronger resilience to device defects than previous methods.

Expanding Capacity with Multilayer Associative Memory

While our hardware-adaptive learning algorithm significantly improves defect tolerance in single-layer Hopfield networks, conventional Hopfield Neural Networks (HNNs) remain fundamentally limited by their single-layer architecture. These traditional implementations suffer from constrained memory capacity and inflexibility due to their mandatory square weight matrix structure. Earlier multilayer or cascaded Hopfield variants[54–56] were mainly designed for binary patterns and, crucially, did not demonstrate capacity gains beyond the linear scaling of single-layer networks or address hardware non-idealities through adaptive training. To overcome these limitations, we extend the HNN framework to a multilayer architecture with recurrent iterations (Figure 4(a)). Because this multilayer formulation relies on inter-layer connections rather than a single symmetric recurrent matrix, conventional single-layer Hopfield learning rules are not directly applicable for learning effective multilayer weights. We therefore employ our loss-decreasing, hardware-adaptive training algorithm to learn stable multilayer weights. Together with the hidden-layer structure, this combination is associated with superlinear capacity scaling beyond classical single-layer Hopfield networks. This design is also consistent with recent theoretical advances showing that overparameterized neural networks can exhibit associative-memory behavior[57,58]. Unlike autoencoders, which are trained in a feedforward manner to learn a compressed representation of a data distribution for generalization, our multilayer Hopfield network is trained to store specific patterns as stable attractors and to retrieve them from corrupted cues via recurrent dynamics; accordingly, the training deliberately fits the target memories and shapes attractor landscape for recall. In the proposed multilayer design, the number of input neurons is the same as the number of output neurons, corresponding to the stored pattern dimension. The output state of iteration n is directly fed as the input to iteration n+1 via a one-to-one mapping of neuron indices. Between the input and the output, one or more hidden layers are inserted to form the multilayer structure. During the associative recall, the recurrent dynamics of the network are preserved through the iterative process: the output of the network is fed back as the input to the next iteration, and all neuron states are updated synchronously. Convergence is guaranteed when the Jacobian operator norm satisfies $\|\mathbf{J}\|_2 < 1$(see Supplementary Note 5), and the process continues until the network converges to a stable point that represents the retrieved memory. In addition, training is performed on a hardware-calibrated model using a mask for stuck-at-fault devices, so the learned weights are directly realizable on non-ideal RRAM crossbars.

To demonstrate the concept, we implemented the multilayer associative memory on our memristor system, using the on-chip crossbar arrays to represent the layers of the architecture. Due to hardware constraints, some multilayer structures exceed the hardware limit in experiments were realized by reprogramming the same array. In our proof-of-concept experiment, we chose a structure with 64 input/output neurons (interfacing with 8×8 patterns) and one hidden layer with 16 hidden neurons. During the training, similar to the single-layer experiment, we trained the system to learn weights for both layers to store 10

MNIST patterns, each randomly selected from a digit. It is noteworthy that this configuration requires only half the synapses of a single-layered HNN. Figures 4(b) and (c) show the experimental readout conductances for weights of the two layers of the multilayer structure, which effectively function as the encoder and decoder, respectively. As demonstrated in Figure 4(d), the system successfully retrieves all patterns despite being presented with inputs corrupted by random 10% pixel flips. Through iterative processing, the multilayer structure achieves perfect pattern recovery, demonstrating that our system can successfully implement multilayer structure for associative memory functionality.

To quantify the benefits of introducing a hidden layer, we conducted a comparison study between multilayer configurations with varying hidden neuron counts and traditional single-layer networks, as shown in Figure 4(e). The results show that system performance can be flexibly tuned by adjusting the number of hidden neurons, with larger hidden layers consistently yielding higher memory capacity. Notably, the multilayer architecture achieves better performance while utilizing fewer synapses, and equivalently fewer memristors compared to conventional Hopfield Neural Networks (HNNs). For example, our implementation with 24 hidden neurons outperforms traditional single-layer networks while requiring only 75% of the synapses. The multilayer design also exhibits a lower probability of spurious states and requires fewer iterations for pattern retrieval, as detailed in the Supplementary Note 6-7. It further demonstrates greater robustness to device non-idealities (Supplementary Figure 7).

It is well established that the capacity of a single-layer HNN scales linearly with the number of neurons. The previous section shows that although our adaptive learning method improves the capacity of the single-layer HNN, it maintains its linear scaling relationship. In contrast, the multilayer structure demonstrates fundamentally improved scalability due to its inherent compression capability. For a fair comparison, we use the number of input neurons N as the reference (i.e., the pattern dimension) and set the number of hidden neurons to $N_h = 0.5N$, so that the number of synapses (required memristor devices) in the multilayer structure matches that of the single-layer HNN since synapses correspond to memristor devices and represent the dominant hardware resource constraint. Figure 4(f) shows the capacity comparison between our multilayer structured model and a traditional single-layer model. For the MNIST dataset, which contains highly correlated patterns, we empirically observe that the capacity of the multilayer structure scales as $\propto N^{1.49}$, significantly exceeding the linear scaling of the single-layer HNN ($\propto N^{1.06}$). Even for uncorrelated random patterns, the multilayer structure still outperforms the single-layer HNN, showing a capacity scaling of $\propto N^{1.11}$ compared to $\propto N^{1.06}$. The advantage of a multilayer structure is less pronounced in this case, as random patterns lack inter-pattern correlations, highlighting that the hidden layer's compression ability is critical for superlinear scalability, a capability traditional single-layer structures lack. Mechanistically, our simulations and analyzes indicate that the multilayer structure performs structure-aware compression that spreads the representations of structured patterns, thereby reducing interference among stored memories. Consistent with this, our analysis shows improved attractor stability in the network dynamics and correspondingly higher capacity. As a result, although our experimental demonstrations focus on a two-layer system, the method naturally extends to deeper architectures. To investigate this scalability, we performed simulations on networks with additional layers, showing that both capacity and attractor stability improve with depth (see Supplementary Note 5 for more details and discussions). Under comparable resource constraints, however, the realized gains depend on the training procedure and the synapse/device budget. A detailed analysis for associative memory with additional layers is provided in Supplementary Note 5.

A significant limitation of the single-layer HNN is their structural inflexibility - once the number of input/output neurons is fixed, both the required number of synapses (memristors) and the system capacity are predetermined. In contrast, multilayer

structures offer the flexibility to modify network architecture by adjusting the number of neurons and layers. We evaluated how multilayer neural network capacity scales with the number of hidden neurons (N_h) when the input/output neuron count remains fixed. As shown in Figure 4(g), with 400 input/output neurons, capacity for random patterns grows linearly ($\propto N_h^{1.05}$), while for MNIST—a dataset with inherent correlations— experimental results indicate superlinear scaling ($\propto N_h^{1.35}$). This enhanced scaling further demonstrates the multilayer architecture's compression capability, which efficiently exploits correlations in complex data. Since the number of synapse N_s of our system is also scales linearly with hidden neurons count $\propto N_h$, which will make the capacity of the system grows linear with synapse count for random patterns ($\propto N_s^{1.05}$) and superlinear for the MNIST dataset ($\propto N_s^{1.35}$). While for the traditional single layer HNN, since the single layer structure lacks flexibility. If you want to increase the capacity, the only way you can do this is to expand the number of neurons, which will make the capacity grow only as $\propto N_s^{\frac{1}{2}}$.

Our multilayer architecture offers practical advantages for hardware implementations, especially when storing only a few high-dimensional patterns. Traditional single-layer HNNs require a fixed number of memristors proportional to N^2, regardless of the number of patterns being stored. In contrast, our multilayer structure provides better resource efficiency. When storing a limited number of patterns, a compact multilayer configuration with fewer hidden neurons can be used. As storage demands increase, the system can be scaled by adding more hidden neurons and corresponding memristors while maintaining the same input/output neuron count (handling the same-sized patterns)- a capability unavailable in conventional HNN architectures. In our design, the required number of memristors is proportional to $N \times N_h$, the number of hidden neurons. Figure 4(h) compares the required memristor device count for an associative memory that stores 20×20 patterns using single-layer and multi-layer (two-layer) network structures. The result shows that the multilayer configuration reduces memristor requirements by 43.7% to 95%, depending on the number of patterns stored in memory.

Continuous Patterns Associative Memory

Beyond its compression capability and superior capacity, our multilayer structure addresses another critical limitation of conventional HNNs for associative memory: the inability to process continuous patterns. Classical single-layer HNNs are effectively restricted to binary patterns because of the sign activation. In our framework, we replace sgn with a differentiable tanh during learning and inference, so that, in principle, the network can operate on continuous patterns, a single-layer HNN remain impractical. The primary limiting factor is that without the compression provided by hidden layers, the highly overlapping nature of continuous patterns leads to severe crosstalk between memory patterns. This causes the system dynamics to become unstable, and usable capacity collapses to only one or a few patterns before retrieval fails (see Supplementary Figure 8). To address this, we couple the adaptive training with the proposed multilayer architecture, enabling the system to effectively associate continuous patterns in the hardware, as illustrated Figure 5(a).

To validate this capability, we implemented the concept in our hardware prototype for continuous pattern association. Figures 5(b) show the mapped conductance weights of both network layers. The retrieval process, illustrated in Fig.5(c), demonstrates successful pattern recovery even when the input state is corrupted by additive Gaussian noise with standard deviation $\sigma = 0.5$ (a relatively strong corruption level for signals in $[-1,1]$), where the corrupted input is generated as $\tilde{\mathbf{x}}^{(p)} = \mathrm{clip}\left(\mathbf{x}^{(p)} + \sigma\boldsymbol{\varepsilon}^{(p)}, -1, 1\right), \quad \boldsymbol{\varepsilon}^{(p)} \sim \mathcal{N}(0,1)$. This result demonstrates the effectiveness of our architecture for continuous-valued pattern recall under substantial cue corruption. Figure 5(d) displays patterns with varying levels of noise alongside the

corresponding retrieved patterns after iterations by both the single-layer HNN and our proposed multilayer structure. Despite the presence of noise, all patterns are accurately retrieved by the multilayer structure, demonstrating its robustness and superior performance. As our analysis shows, this resilience is maintained across a broad noise range, with retrieval precision only degrading noticeably as the noise standard deviation approaches 0.5 (see Supplementary Figure 9). In contrast, the single-layer HNN fails to retrieve the correct patterns at every noise level since of the limited capacity (Fig. 5d), Moreover, our method exhibits superior robustness to device non-idealities compared with the single-layer design, as shown in Supplementary Figure 10. Figure 5(e) illustrates the retrieval similarity as a function of iteration steps. It is evident that, after sufficient iterations, all patterns are successfully retrieved and remain stable, confirming the effectiveness of the retrieval process of our proposed multilayer structure. Our analysis shows that the system's Jacobian has a spectral norm less than one ($\|\mathbf{J}\|_2 < 1$), which guaranties convergence to a fixed-point attractor (see Supplementary Note 5 for the full analysis).

To ensure consistency with the binary case, we define storage capacity for continuous patterns as the maximum number of patterns that can be retrieved with a cosine similarity exceeding 0.99 when the input is corrupted by additive Gaussian noise with standard deviation 0.3, which yields an average cue–target cosine of ≈ 0.90, matching the perturbation introduced by 5% bit flips in the binary experiments. Figure 5(f) shows this capacity as a function of input dimensionality, with hidden neuron dimension set to half that of the original input patterns for comparison purposes. We empirically observe that the capacity for continuous patterns exceeds linear scaling ($\propto N$), achieving superlinear scalability of $\propto N^{1.74}$ for MNIST and $\propto N^{1.66}$ for random patterns. Similar to the multilayer structures for binary patterns, increasing the hidden dimension enhances the system's capacity, underscoring the scalability of the multilayer structure. Figure 5(g) demonstrates the relationship between system capacity and hidden layer dimension. The results reveal that capacity scales with hidden dimension as $\propto N^{0.67}$ for MNIST patterns and $\propto N^{0.53}$ for random patterns. This scaling behavior confirms that the system's capacity can be effectively tuned by adjusting the number of hidden neurons, providing valuable flexibility for practical implementations. The stronger scaling observed for MNIST (0.67 vs 0.53) further highlights the architecture's enhanced efficiency when processing structured, correlated data. The observed difference in scaling between the binary and continuous cases comes from the more demanding nature of the continuous pattern task. Reproducing precise continuous values, rather than only the correct signs, imposes stricter training and makes forming high-fidelity attractors more difficult, thus reducing the capacity gained per hidden neuron.

Highly Non-idealities Resilient and Efficient System

In analog computing, computation is affected by inevitable noise. In our system, the most significant non-idealities arise from two main sources: the mismatch between the target and mapped conductance values, and noise introduced by peripheral circuitry, such as ADC noise and line resistance. Both of these factors affect the precision of weights in the analog neural network. These noise sources manifest as deviations between the readout conductance and the intended target conductance. Figure 6(a) illustrates the experimental distribution of these conductance differences during the mapping of a weight matrix to our hardware, with the mapped conductance range spanning 0 to 150 μS. The conductance difference follows a Gaussian distribution with a near-zero mean (0.108 μS) and a standard deviation of 3.894 μS. Using this measured conductance variation, we simulate the effect of hardware non-idealities on experimental performance. Figure 6(b) presents the cosine similarity as a function of conductance variation for the previously described hardware-implemented associative memory tasks. Our system demonstrates strong robustness to such non-idealities: for continuous-valued patterns, the similarity experiences only

a minor decline under the observed experimental variations, and for binary applications, the pattern retrievals were mostly perfect because of the binarization from the sign function. This robustness to device drift has significant implications for the system-level overhead of our hardware-adaptive approach. For moderately stable devices, like those used in this work, the resilience allows for very infrequent characterization scans (e.g., on the order of hours or days), and our analysis shows that the corresponding energy and latency overhead is negligible, contributing less than 0.1% to the system's operational cost. If more stable devices exist in the future, our method simplifies to a one-time calibration. Therefore, our hardware-adaptive method provides a practical and efficient solution across a realistic range of device stabilities (see Supplementary Note 8 for detailed analysis).

By leveraging the fully parallel update capability of memristor-based computing hardware, our method achieves significantly improved energy efficiency and reduced latency for associative pattern recall, compared to conventional HNN implementations for associative memory, which can only update the node states asynchronously[35–40]. Figure 6(c) presents the energy consumption results (see Supplementary Note 9 for detailed analysis). Compared to previous asynchronous implementations, our single-layer structure achieves 2.68× higher energy efficiency when storing two patterns in the memory, representing the case where only a few patterns are stored, 2.76× higher energy efficiency when 10 patterns are stored, representing more patterns are stored in the system. The experiments were done based on a system that stores 64-dimensional patterns, and the improvement is expected to grow with larger systems. Additionally, the multilayer structure demonstrates 8.8× and 7.01× improvements in energy efficiency over asynchronous single-layer associative memory for fewer (2 patterns) and more stored patterns (10 patterns). Regarding latency, as shown in Figure 6(d) (see Supplementary Note 9 for detailed analysis), our method reduces latency by 99.4% with fewer patterns (2 patterns) for both single-layer HNN and multi-layer structure. When the storage is larger (10 patterns), we observe a 99.4% reduction and an 99.7% reduction over asynchronous implementations, owing to the full utilization of parallelism in analog computing.

Discussion

In summary, we develop a hardware-adaptive learning algorithm that addresses three major limitations of prior memristor-based Hopfield-style associative memory systems: sensitivity to hardware imperfections, limited scalability/flexibility, and restricted support for binary-only patterns. We experimentally validate the proposed training framework on an integrated memristor crossbar associative recall system, and demonstrate significantly improved capacity and defect tolerance compared with state-of-the-art methods. On the MNIST dataset, it achieves double the capacity of existing algorithms, and triple the capacity of the SOTA baseline with a 50% stuck-at fault ratio. Building on the single-layer design, we further extend the system to a multilayer architecture, which is found to improves both capability and flexibility. The introduction of a hidden layer enables superlinear scaling ($\propto N^{1.49}$) for correlated patterns, significantly outperforming the linear scaling ($\propto N^{1.06}$) in single-layer networks. The hidden layer also adds flexibility and improves memristor usage efficiency (43.75% to 95% reduced memristor usage) compared to single-layer Hopfield Neural Networks (HNNs). The multilayer architecture not only increases capacity but also extends the functionality of associative memory systems to support continuous patterns, while also showing superlinear scalability. This makes the system more applicable to real-world scenarios, where data often exists in continuous form. By leveraging the inherent parallelism of memristor arrays and using synchronous state updates, our method achieves a 2.68× to 2.76× improvement in energy efficiency and a 99.4% reduction in processing time compared to previous asynchronous updates.

These metrics are further improved due to the enhanced capabilities of the multilayer structure, achieving 2.53× to 3.28× better energy efficiency and 1–2× faster performance compared to single-layer implementations. By overcoming key limitations and harnessing the unique properties of memristors, our method demonstrates the strong potential of adaptive learning and multilayer architectures for associative memory and neuromorphic computing. Future work will explore even more diverse and complex neural network structures for associative memory and investigate their integration into broader computational frameworks and neuromorphic devices.

Methods

Memristor Integration

The memristor devices in this platform were integrated onto foundry CMOS selector and periphral circuits (with a standard 180 nm technology node) using a proprietary back-end process. The process began by removing the chip's passivation layer, followed by patterning a 2 nm layer of chromium (Cr) and a 10 nm layer of platinum (Pt) to form the bottom electrode. Next, a switching layer of 4–8 nm tantalum oxide (TaOx) was deposited via reactive sputtering. A 10 nm layer of tantalum (Ta) was then sputter-deposited as the top electrode, with an additional 10 nm platinum (Pt) layer added for protection. More details on the fabrication process can be found in Ref[48].

Programming Strategy

In this work, we employ a write-and-verify programming strategy to adjust the memristor conductance to the desired target value. Initially, we define the target conductance and set a programming error tolerance range of 5 μS. The target conductance is derived from the original matrix and scaled to the desired conductance range (0 μS to 150 μS) without quantization. During the write-and-verify process, the conductance of each device in the memristor array is first measured using a 0.2 V read voltage. If the measured conductance falls within the defined tolerance range of the target conductance, no further action is taken for that memristor. However, if the conductance deviates from the target by more than the allowable tolerance, a RESET or SET voltage is applied to decrease or increase the conductance, respectively, bringing it closer to the target value. Throughout the programming iterations, both the RESET and SET voltages, along with the gate voltage, are gradually increased.

Hardware-Adaptive Training Algorithm

For the single-layer structure, the L2 distance is used to measure the discrepancy between the retrieved patterns and the original patterns in the system. To simplify the calculations, we set the parameter $\lambda = 1$, which affects only the steepness of the objective function. Consequently, the final loss function is defined as:

$$\text{Loss} = \frac{1}{P}\sum_{p}\left(\mathbf{x}^{(p)} - \tanh\left(\mathbf{W}\mathbf{x}^{(p)} - \mathbf{b}\right)\right)^2 \tag{5}$$

During training, we set the learning rate to 3×10^{-2}, and run the process for a maximum of 10,000 steps. Training terminates early if the loss drops below 1×10^{-8}.

For the multi-layer structure, the training loss function is modified to:

$$\text{Loss} = \frac{1}{P}\sum_{p}\left(\mathbf{x}^{(p)} - \mathbf{f}\left(\mathbf{x}^{(p)}\right)\right)^2. \tag{6}$$

Here, $\mathbf{f}\left(\mathbf{x}^{(p)}\right)$ denotes the multilayer network's output.The learning rate (lr) is generally set to 3×10^{-4} with a maximum of 60,000 epochs, and employ early stopping when the training loss falls below 10^{-8}. In some cases (unstructured random data), we fine-tune the hyperparameters to epochs = 100,000 and lr $= 2 \times 10^{-4}$. Training is performed with the RMSprop optimizer to minimize the loss. During training, we use Root Mean Square Propagation (RMSprop) as the optimizer to reduce the loss. To handle stuck-at-fault devices, we make the weights at the corresponding locations untrainable and set their values to zero during training.

Capacity definition

In order to evaulate the performance of the differet methods and model for the performance of associative task. Throughout the manuscript we define capacity as the maximum number of patterns that can be successfully retrieved when the network is inputted with a corrupted version of each pattern. Specifically, for binary patterns we corrupt the input by flipping 5% of the bits (expected cosine similarity of the cue to the target ≈0.90) and declare successful retrieval when the final recovered state has cosine similarity ≥ 0.99 with the target. Compared with the previous (noiseless) definition[53], our criterion yields lower numerical capacity. This is expected, because our evaluation is stricter: it explicitly probes the basin of attraction by requiring recovery from corrupted cues, rather than merely checking whether patterns are fixed points. In this sense, our measure is more appropriate for robustness under noise.For continuous patterns, we add the Gaussian noise of standard deviation 0.3 to make consistent with binary case, so that the average cue-to-target cosine is ≈ 0.90—matching the perturbation induced by 5% bit flips in the binary case.

PCB-based interface board with memristor-chip

A custom-designed printed circuit board (PCB) supports both programming and pattern retrieval operations on the chip. The PCB serves as an interface between the chip and external off-the-shelf components. This interface board includes a microcontroller that enables communication with a computer, as well as a power supply that delivers the necessary operating voltage via power supply or on-board digital-to-analog converters (DACs). Most peripheral circuits, such as transimpedance amplifiers and analog-to-digital converters (ADCs), are integrated on-chip, while the board mainly provides probes for troubleshooting purposes. The chip itself is wire-bonded to a chip-holder on the testing board.

Data availability

All data supporting this study and its findings are available within the article, its Supplementary Information and associated files. Source data have been deposited in Figshare at:

https://doi.org/10.6084/m9.figshare.31266745

Code availability

All the necessary codes used in the tactile experiments and visual experiments and their descriptions are available in Github: https://github.com/hecp2025/SuperlinearAssociativeMemory. The version used for this study has been archived in Zenodo: https://doi.org/10.5281/zenodo.18494623.

References

1. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *nature* **521**, 436–444 (2015).
2. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778 (2016).
3. Vaswani, A. *et al.* Attention is all you need. *Adv. neural information processing systems* **30** (2017).
4. Pavlov, P. I. Conditioned reflexes: an investigation of the physiological activity of the cerebral cortex. *Annals neurosciences* **17**, 136 (2010).
5. Pagiamtzis, K. & Sheikholeslami, A. Content-addressable memory (cam) circuits and architectures: A tutorial and survey. *IEEE journal solid-state circuits* **41**, 712–727 (2006).
6. Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. national academy sciences* **79**, 2554–2558 (1982).
7. Hopfield, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. national academy sciences* **81**, 3088–3092 (1984).
8. Hopfield, J. J. & Tank, D. W. "neural" computation of decisions in optimization problems. *Biol. cybernetics* **52**, 141–152 (1985).
9. Hopfield, J. J. & Tank, D. W. Computing with neural circuits: A model. *Sci.* **233**, 625–633 (1986).
10. Krotov, D. & Hopfield, J. J. Dense associative memory for pattern recognition. *Adv. neural information processing systems* **29** (2016).
11. Demircigil, M., Heusel, J., Löwe, M., Upgang, S. & Vermet, F. On a model of associative memory with huge storage capacity. *J. Stat. Phys.* **168**, 288–299 (2017).
12. Krotov, D. A new frontier for hopfield networks. *Nat. Rev. Phys.* **5**, 366–367 (2023).
13. Goto, H., Tatsumura, K. & Dixon, A. R. Combinatorial optimization by simulating adiabatic bifurcations in nonlinear hamiltonian systems. *Sci. advances* **5**, eaav2372 (2019).
14. Strukov, D. B., Snider, G. S., Stewart, D. R. & Williams, R. S. The missing memristor found. *nature* **453**, 80–83 (2008).
15. Rao, M. *et al.* Thousands of conductance levels in memristors integrated on cmos. *Nat.* **615**, 823–829 (2023).
16. Sharma, D. *et al.* Linear symmetric self-selecting 14-bit kinetic molecular memristors. *Nat.* **633**, 560–566 (2024).
17. Li, C. *et al.* Analogue signal and image processing with large memristor crossbars. *Nat. electronics* **1**, 52–59 (2018).
18. Sheridan, P. M. *et al.* Sparse coding with memristor networks. *Nat. nanotechnology* **12**, 784–789 (2017).

19. Zhang, W. *et al.* Edge learning using a fully integrated neuro-inspired memristor chip. *Sci.* **381**, 1205–1211 (2023).

20. Li, C. *et al.* Long short-term memory networks in memristor crossbar arrays. *Nat. Mach. Intell.* **1**, 49–57 (2019).

21. Wang, Z. *et al.* In situ training of feed-forward and recurrent convolutional memristor networks. *Nat. Mach. Intell.* **1**, 434–442 (2019).

22. Ambrogio, S. *et al.* Equivalent-accuracy accelerated neural-network training using analogue memory. *Nat.* **558**, 60–67 (2018).

23. Ambrogio, S. *et al.* An analog-ai chip for energy-efficient speech recognition and transcription. *Nat.* **620**, 768–775 (2023).

24. Wan, W. *et al.* A compute-in-memory chip based on resistive random-access memory. *Nat.* **608**, 504–512 (2022).

25. Zidan, M. A., Strachan, J. P. & Lu, W. D. The future of electronics based on memristive systems. *Nat. electronics* **1**, 22–29 (2018).

26. Yao, P. *et al.* Fully hardware-implemented memristor convolutional neural network. *Nat.* **577**, 641–646 (2020).

27. Feng, Y. *et al.* Memristor-based storage system with convolutional autoencoder-based image compression network. *Nat. Commun.* **15**, 1132 (2024).

28. Jiang, M., Shan, K., He, C. & Li, C. Efficient combinatorial optimization by quantum-inspired parallel annealing in analogue memristor crossbar. *Nat. Commun.* **14**, 5927 (2023).

29. Cai, F. *et al.* Power-efficient combinatorial optimization using intrinsic noise in memristor hopfield neural networks. *Nat. Electron.* **3**, 409–418 (2020).

30. Yang, K. *et al.* Transiently chaotic simulated annealing based on intrinsic nonlinearity of memristors for efficient solution of optimization problems. *Sci. advances* **6**, eaba9901 (2020).

31. Zidan, M. A. *et al.* A general memristor-based partial differential equation solver. *Nat. Electron.* **1**, 411–420 (2018).

32. Le Gallo, M. *et al.* Mixed-precision in-memory computing. *Nat. Electron.* **1**, 246–253 (2018).

33. Jiang, H. *et al.* A provable key destruction scheme based on memristive crossbar arrays. *Nat. Electron.* **1**, 548–554 (2018).

34. Wang, Z., Wu, Y., Park, Y. & Lu, W. D. Safe, secure and trustworthy compute-in-memory accelerators. *Nat. Electron.* 1–12 (2024).

35. Eryilmaz, S. B. *et al.* Brain-like associative learning using a nanoscale non-volatile phase change synaptic device array. *Front. neuroscience* **8**, 205 (2014).

36. Li, Y., Wang, S., Yang, K., Yang, Y. & Sun, Z. An emergent attractor network in a passive resistive switching circuit. *Nat. Commun.* **15**, 7683 (2024).

37. Wang, Y., Yu, L., Wu, S., Huang, R. & Yang, Y. Memristor-based biologically plausible memory based on discrete and continuous attractor networks for neuromorphic systems. *Adv. Intell. Syst.* **2**, 2000001 (2020).

38. Yan, M. *et al.* Ferroelectric synaptic transistor network for associative memory. *Adv. Electron. Mater.* **7**, 2001276 (2021).

39. Hu, S. *et al.* Associative memory realized by a reconfigurable memristive hopfield neural network. *Nat. communications* **6**, 7522 (2015).

40. Zhou, Y. *et al.* Associative memory for image recovery with a high-performance memristor array. *Adv. Funct. Mater.* **29**, 1900155 (2019).

41. Pedretti, G. *et al.* A spiking recurrent neural network with phase-change memory neurons and synapses for the accelerated solution of constraint satisfaction problems. *IEEE J. on Explor. Solid-State Comput. Devices Circuits* **6**, 89–97 (2020).

42. Do, H. The organization of behavior. *New York* (1949).

43. Storkey, A. J. & Valabregue, R. The basins of attraction of a new hopfield learning rule. *Neural Networks* **12**, 869–876 (1999).

44. Kanter, I. & Sompolinsky, H. Associative recall of memory without errors. *Phys. Rev. A* **35**, 380 (1987).

45. Tolmachev, P. & Manton, J. H. New insights on learning rules for hopfield networks: Memory and objective function minimisation. In *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–8 (IEEE, 2020).

46. Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R. & Bengio, Y. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830* (2016).

47. Paren, A. & Poudel, R. P. Training binarized neural networks the easy way. In *BMVC*, 35 (2022).

48. Sheng, X. *et al.* Low-conductance and multilevel cmos-integrated nanoscale oxide memristors. *Adv. electronic materials* **5**, 1800876 (2019).

49. Zoppo, G., Marrone, F. & Corinto, F. Equilibrium propagation for memristor-based recurrent neural networks. *Front. neuroscience* **14**, 501774 (2020).

50. Yi, S.-i., Kendall, J. D., Williams, R. S. & Kumar, S. Activity-difference training of deep neural networks using memristor crossbars. *Nat. Electron.* **6**, 45–51 (2023).

51. McCloskey, M. & Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, vol. 24, 109–165 (Elsevier, 1989).

52. Robins, A. & McCALLUM, S. Catastrophic forgetting and the pseudorehearsal solution in hopfield-type networks. *Connect. Sci.* **10**, 121–135 (1998).

53. Amit, D. J., Gutfreund, H. & Sompolinsky, H. Statistical mechanics of neural networks near saturation. *Annals physics* **173**, 30–67 (1987).

54. Jagota & Jakubowicz. Knowledge representation in a multilayered hopfield network. In *International 1989 Joint Conference on Neural Networks*, 435–442 (IEEE, 1989).

55. Jin, L., Nikiforuk, P. & Gupta, M. On the multilayered hopfield neural networks. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, vol. 3, 1443–1448 (IEEE, 1994).

56. Young, S. S., Scott, P. D. & Nasrabadi, N. M. Object recognition using multilayer hopfield neural network. *IEEE Transactions on Image Process.* **6**, 357–372 (1997).

57. Zhang, C., Bengio, S., Hardt, M., Mozer, M. C. & Singer, Y. Identity crisis: Memorization and generalization under extreme overparameterization. *arXiv preprint arXiv:1902.04698* (2019).

58. Radhakrishnan, A., Belkin, M. & Uhler, C. Overparameterized neural networks implement associative memory. *Proc. Natl. Acad. Sci.* **117**, 27162–27170 (2020).

Acknowledgements

This work was supported in part by the Research Grant Council of Hong Kong SAR (17207925(C.L.), C7003-24Y(C.L.), C1009-22GF(C.L.), T45-701/22-R(C.L.), C500124Y(C.L.)), Innovation and Technology Commission of Hong Kong SAR (MHP/363/24(C.L.)), MOST (2024YFE0217000(C.L.)), ACCESS – an InnoHK center by ITC(C.L.), and Croucher Innovation Award from the Croucher Foundation(C.L.).

Author Contributions

C.H. and C.L. conceived and designed the study. C.H. performed the experiments and collected the hardware measurements, developed the training framework, carried out simulations, and analyzed the results. M.J., K.S., S.-H.Y., and Z.L. contributed to technical discussions and project development. S.W. assisted with figure preparation and data visualization. G.P. and J.I. provided hardware/platform support and contributed to technical discussions. C.L. supervised the project. C.H. and C.L. wrote the manuscript with input from G.P., J.I. All the authors discussed the results and commented on the manuscript.

Competing Interests

The authors declare no competing interests.

Figure 1. Key concepts of our high-capacity memristor-based associative memory enabled by hardware-adaptive learning: (a) Associative memory in the human brain and implemented by Hopfield neural networks (HNN). The human brain's associative memory retrieves complete information from incomplete inputs. Similarly, an artificial HNN can recover stored patterns from corrupted or partial inputs. (b) A schematic showing the digital von-Neumann architecture computing hardware with separation of the processing unit, control unit, and memory, leading to higher latency and lower energy efficiency when implementing HNNs. (c) Memristor-based analog in-memory computing hardware leverages Ohm's law and Kirchhoff's circuit laws to perform efficient computations, achieving higher energy efficiency and lower latency for HNNs. (d) Compared with previous approaches that relied on offline training, single-layer structures, and binary patterns, the hardware-adaptive learning method proposed in this work enables higher capacity and defect tolerance , with the support of multilayer architectures and works with both binary and continuous patterns.

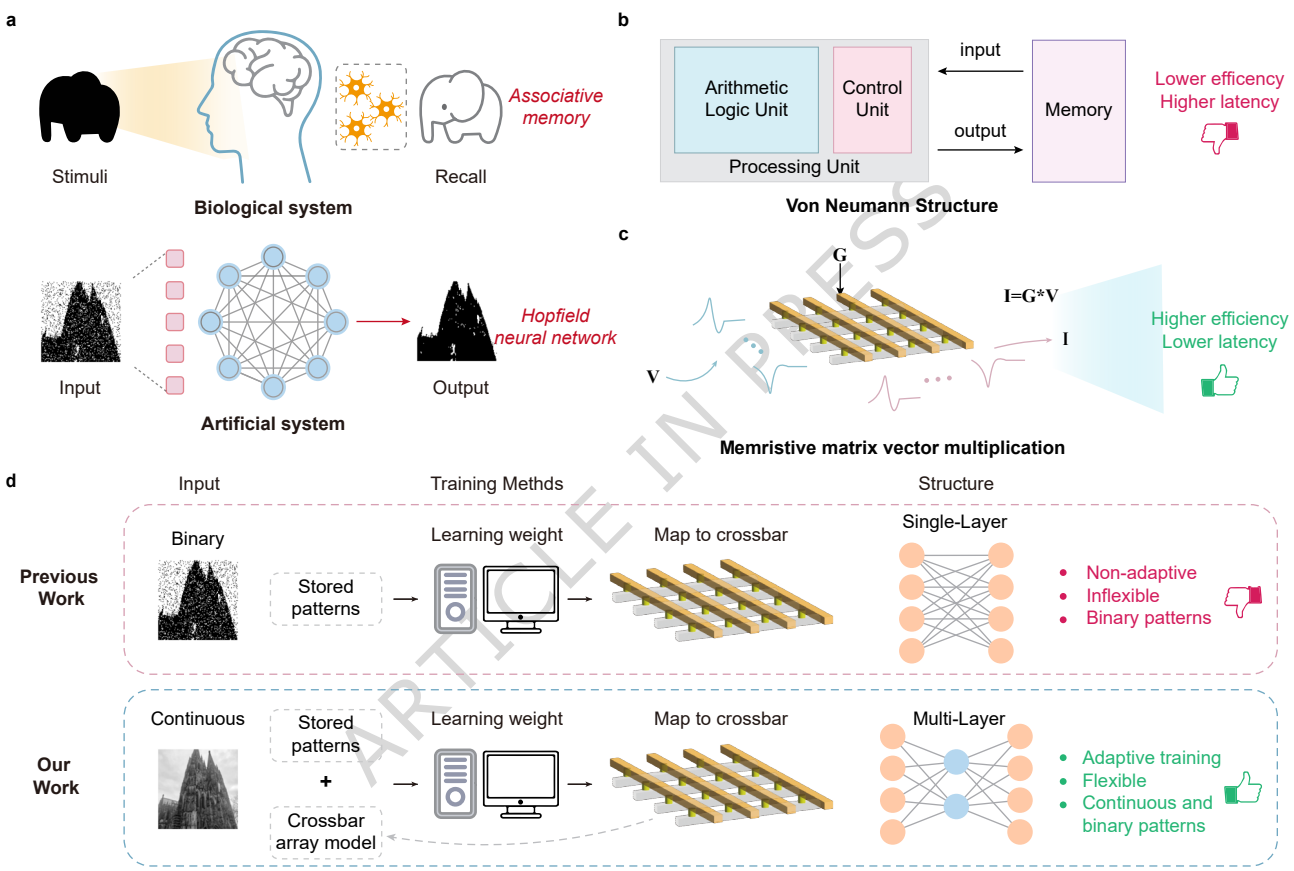
Figure 2. Integrated memristor hardware for associative memory experiments : (a) Photo of our memristor-based associative memory system, comprising our integrated memristor chip, a power supply, a microcontroller, and peripheral circuitry for troubleshooting. (b) The layout of our integrated memristor chip with three 64 $\times$ 64 crossbar arrays alongside fully integrated peripheral circuits—including transimpedance amplifiers (TIAs), multiplexers (MUXs), analog-to-digital converters (ADCs), and drivers. Memristor devices were fabricated via back-end-of-line (BEOL) processing in an in-house cleanroom, while CMOS peripheral circuits were fabricated using TSMC's 180 nm technology. (c) Optical microscopy image of one of the 64 $\times$ 64 memristor crossbar arrays. (d) DC I-V characteristics of a one-transistor-one-memristor (1T1M) device over 100 cycles, with the dark line showing the average value. (e) Linearity analysis of devices programmed to different conductance states, demonstrating excellent I-V linearity for accurate analog computing. (f) Retention test for 16 conductance levels (0-150 µS, 10 µS one step), with each level tested across 256 devices. The dark line represents the median retention, and the shaded region denotes the interquartile range, confirming stable conductance beyond 10^5 s.

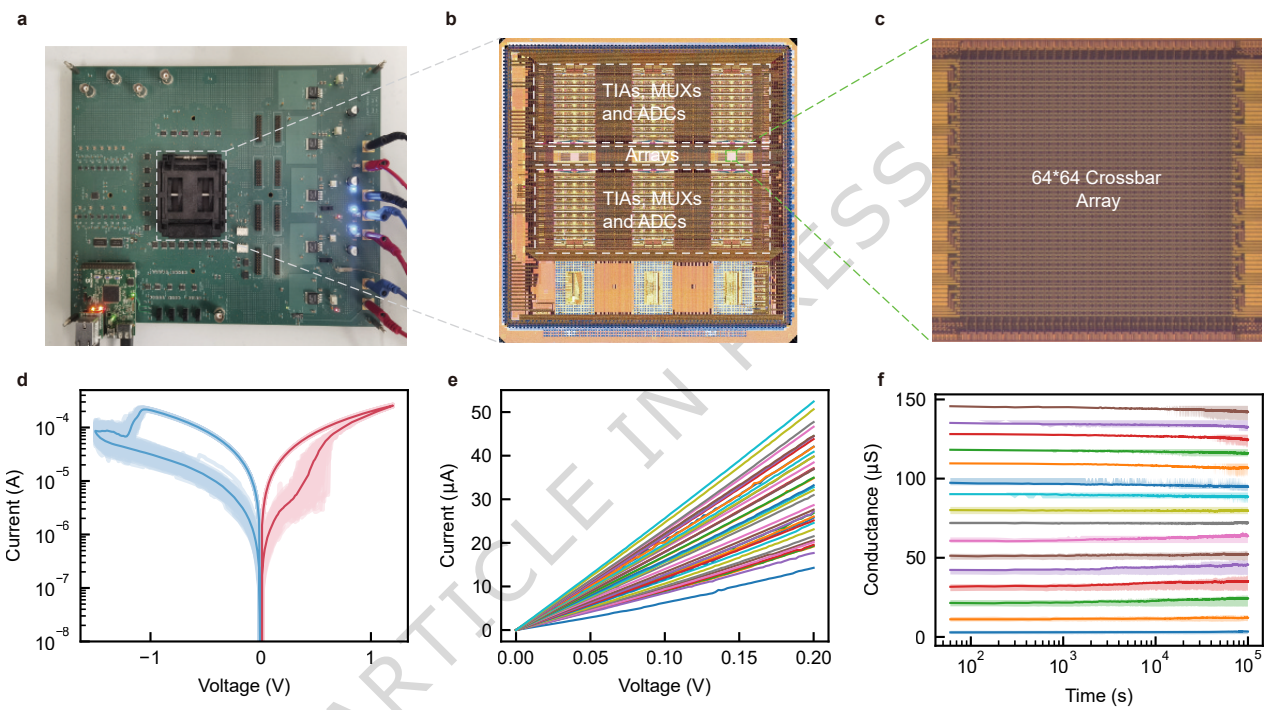
Figure 3. Experimental demonstration of adaptive learning algorithm for single-layer HNN and performance analysis (a) Overview of the adaptive training and retrieval flow. A physics-based crossbar model is used to train weights while accounting for hardware non-idealities (e.g., stuck-at faults). During inference, corrupted inputs are iteratively updated until convergence and read out as the final state. (b) Target conductance values representing the learned weights to be stored in the memristor array. (c) Experimental readout conductance values from the memristor system after programming the target values, showing close matches with (b). (d) Comparison between expected and experimental analog computing results from the crossbar, demonstrating high computing accuracy. (e) Examples of corrupted input patterns (with intentionally added noise), the corresponding stable states retrieved by the memristor system, and the originally stored patterns (ground truth). (f) Energy (blue) decreases and cosine similarity (red) increases monotonically with each iteration, eventually converging to stable equilibrium values. (g) Our algorithm retrieves patterns with better quality, as quantified by cosine similarity, outperforming previous methods, including the state-of-the-art (SOTA) pseudoinverse approach. Shaded regions indicate the interquartile range over 20 runs.Solid lines indicate the mean. EP denotes equilibrium propagation; SOTA denotes the pseudoinverse method. (h) As the fault ratio exceeds a threshold, the quality of retrieved patterns (quantized by cosine similarity) degrades dramatically, but our method exhibits superior defect tolerance compared to SOTA. (i) System capacity versus the number of neurons, where capacity is defined as the maximum number of patterns that can be retrieved with a cosine similarity greater than 0.99. Larger neuron counts yield higher capacity. Compared to the state-of-the-art, our method achieves approximately double the capacity. (j) System capacity decreases with more stuck-at-fault devices (400 neurons), yet our method maintains a higher capacity, demonstrating better defect tolerance.

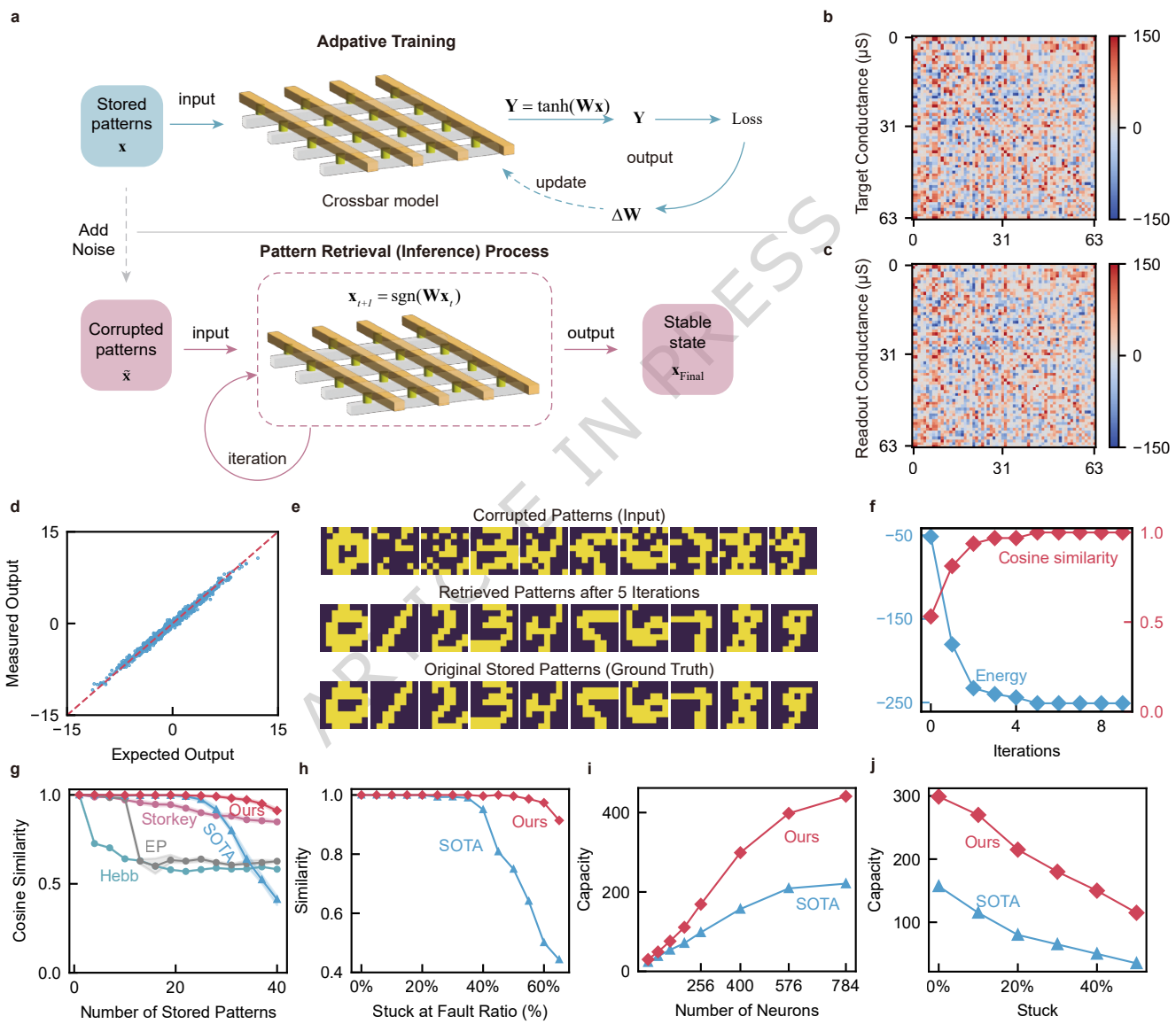
Figure 4. Associative memory implemented with multilayer structure: (a) Schematic of the Hopfield Neural Network (HNN) for associative memory with a traditional one-layer fully-connected structure and the multilayer structure enabled by our adaptive learning method. (b) Experimental readout conductance of the 1st layer and (c) the 2nd layer in a two-layer system. (d) Corrupted patterns serve as the input to the associative memory (generated by flipping 10% of the pixels in the stored patterns), the experimentally retrieved patterns, and the original stored patterns that serve as the ground truth. (e) Performance comparison between the multilayer structure (with varying numbers of hidden neurons, e.g., $N_h = 16$) and the single-layered HNN. The vertical axis represents the cosine similarity between stored and retrieved patterns, while the horizontal axis indicates the number of stored patterns. Shaded regions indicate the inter-quartile range over 20 runs; solid lines indicate the mean. (f) Capacity comparison between single-layer HNN and multilayer structures (where the hidden layer dimension is half the input neuron count to make the synapse number the same as that in single-layer networks) across varying input neuron numbers. (g) Capacity of the multilayer neural network as a function of hidden neuron count, with a fixed input size of 400 neurons. (h) Required memristor count for storing different numbers of 20×20 patterns (capacity) in single-layer and multilayer structures with 400 neurons.

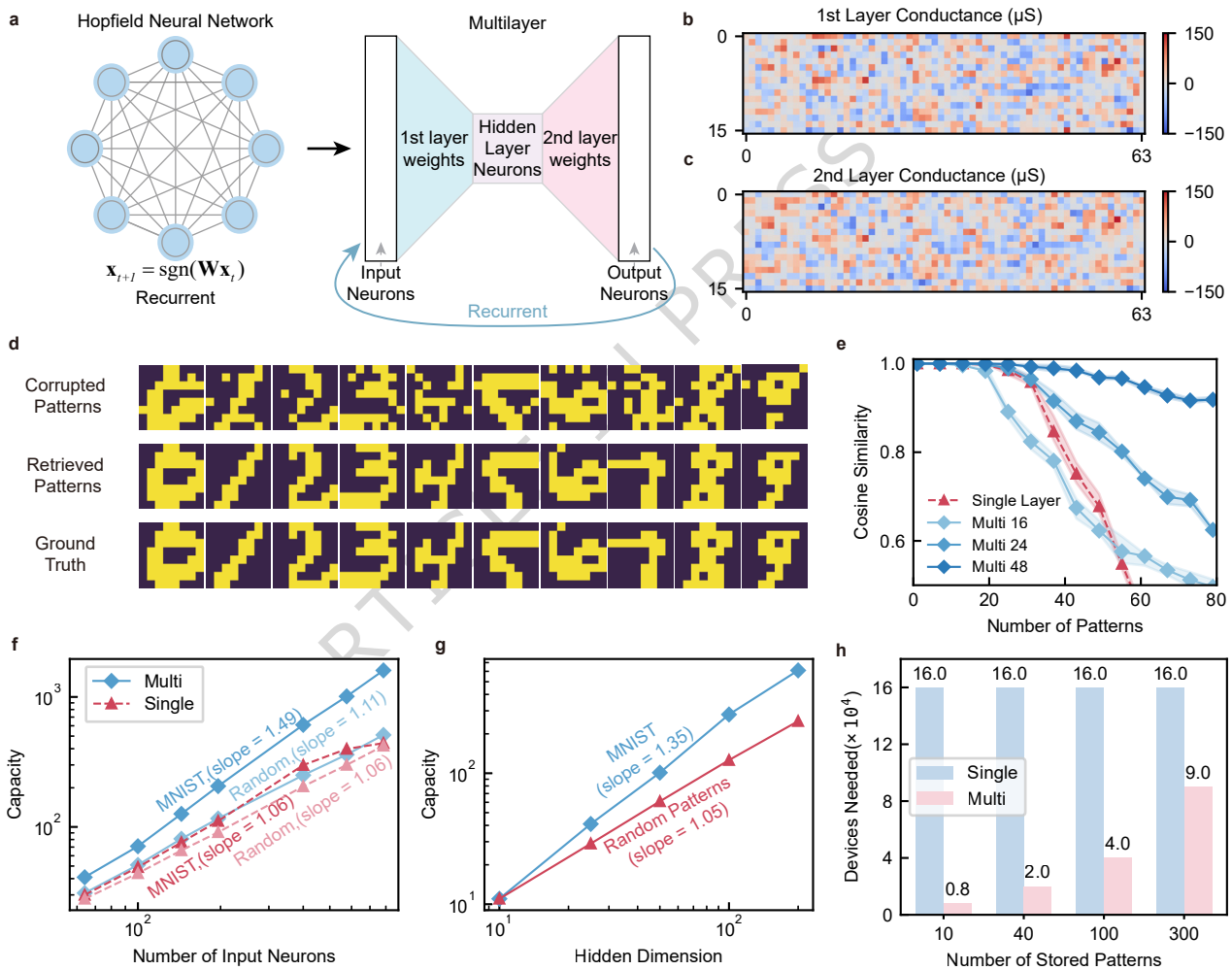
Figure 5. Continuous patterns associative memory enabled by adaptive learning and multiplayer network: (a) Schematic comparison between conventional HNN that supports only binary patterns and the multilayer associative memory with adaptive learning proposed in this work that supports both binary and continuous patterns. (b) Experimental readout weights (each represented by the conductance difference of two adjacent memristors) of the 1st and 2nd layers in a two-layer structure. (c) Hardware implementation results for associative memory with continuous patterns, where corrupted patterns are generated by adding Gaussian noise (amplitude = 0.6) to the original patterns. (d) Input patterns with varying noise levels (0.4, 0.5, 0.6) and their corresponding retrieved patterns by single-layer HNN and multilayer structure. While the multilayer associative memory correctly retrieved the pattern, the single-layer one barely worked. (e) Cosine similarity between stored and retrieved patterns as a function of iteration. (f) Memory capacity as a function of the number of input neurons for continuous patterns, where the number of hidden neurons is set to half the number of input/output neurons. (g) Memory capacity as a function of the number of hidden neurons for continuous patterns, with a fixed total of 400 neurons.

Figure 6. Performance Benchmarks: (a) The typical mean and standard deviation of the programming error between the target conductance (ranging from 0 to 150 μS) and the actual readout conductance from our integrated memristor crossbar. (b) Simulation results showing the impact of conductance variation on performance. The vertical dotted line indicates our experimental conductance variation. The experiments utilize a single-layer and multilayer structure to store 10 binary or continuous patterns, following the same setup as the previously described hardware implementation of associative memory experiments. Shaded regions indicate the inter-quartile range over 20 runs; solid lines indicate the median. (c) Energy consumption and (d) latency of hardware implementations using different node update schemes (asynchronous updates - previous work only implemented asynchronous updates, synchronous updates), and network structure (single layer and multilayer) when storing different numbers of patterns (2 and 10). The evaluation is based on a 64-neuron system.






a
Adpative Training
Stored patterns x
input
Y = tanh(Wx)
Y
Loss
output
update
ΔW
Crossbar model
Add Noise
Pattern Retrieval (Inference) Process
Corrupted patterns x̃
input
x_{t+1} = sgn(Wx_t)
output
Stable state x_Final
iteration
b
Target Conductance (μS)
c
Readout Conductance (μS)
d
Measured Output
Expected Output
e
Corrupted Patterns (Input)
Retrieved Patterns after 5 Iterations
Original Stored Patterns (Ground Truth)
f
Cosine similarity
Energy
Iterations
g
Cosine Similarity
Storkey
Ours
EP
Hebb
SOTA
Number of Stored Patterns
h
Similarity
Ours
SOTA
Stuck at Fault Ratio (%)
i
Capacity
Ours
SOTA
Number of Neurons
j
Capacity
Ours
SOTA
Stuck


a
Hopfield Neural Network
$\mathbf{x}_{t+1} = \mathrm{sgn}(\mathbf{W}\mathbf{x}_t)$
Recurrent
Multilayer
1st layer weights
Hidden Layer Neurons
2nd layer weights
Input Neurons
Output Neurons
Recurrent
b
1st Layer Conductance (μS)
c
2nd Layer Conductance (μS)
d
Corrupted Patterns
Retrieved Patterns
Ground Truth
e
Cosine Similarity
Number of Patterns
Single Layer
Multi 16
Multi 24
Multi 48
f
Capacity
Number of Input Neurons
Multi
Single
MNIST,(slope = 1.49)
Random,(slope = 1.11)
MNIST,(slope = 1.06)
Random,(slope = 1.06)
g
Capacity
Hidden Dimension
MNIST (slope = 1.35)
Random Patterns (slope = 1.05)
h
Devices Needed(× 10^4)
Number of Stored Patterns
Single
Multi
16.0
0.8
2.0
4.0
9.0