nature machine intelligence

SUPPLEMENTARY INFORMATION

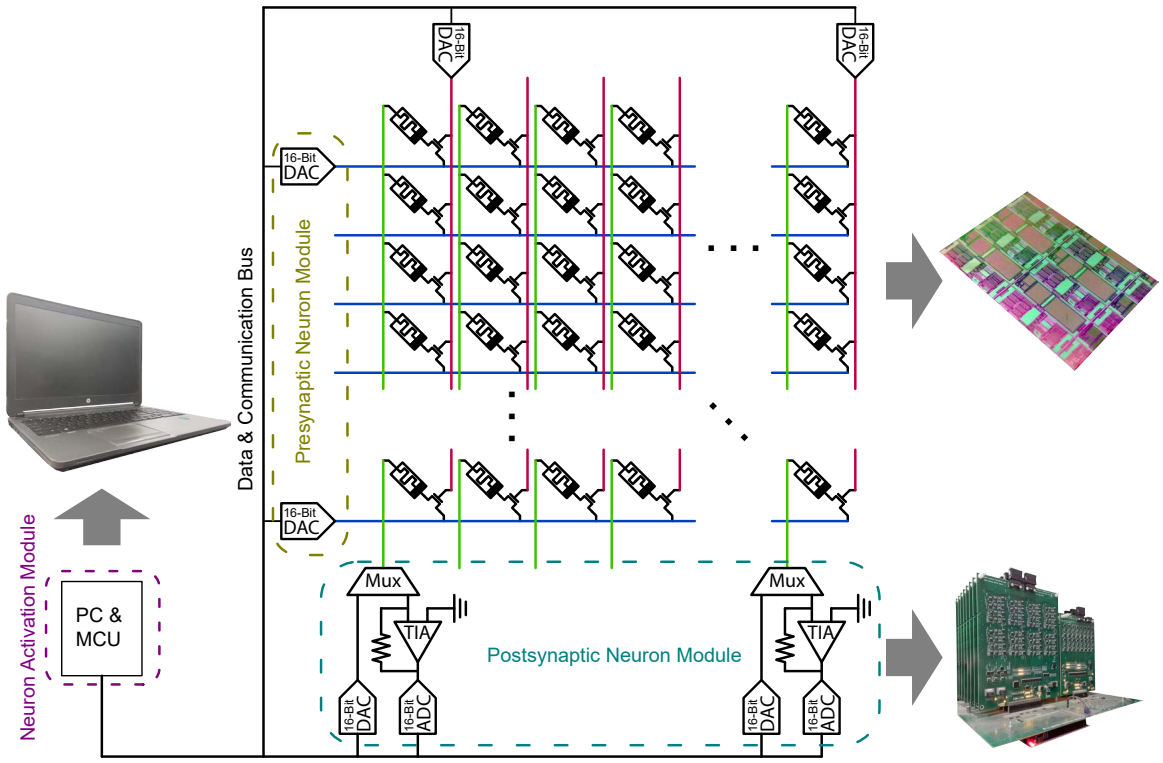
https://doi.org/10.1038/s42256-019-0089-1

In the format provided by the authors and unedited.

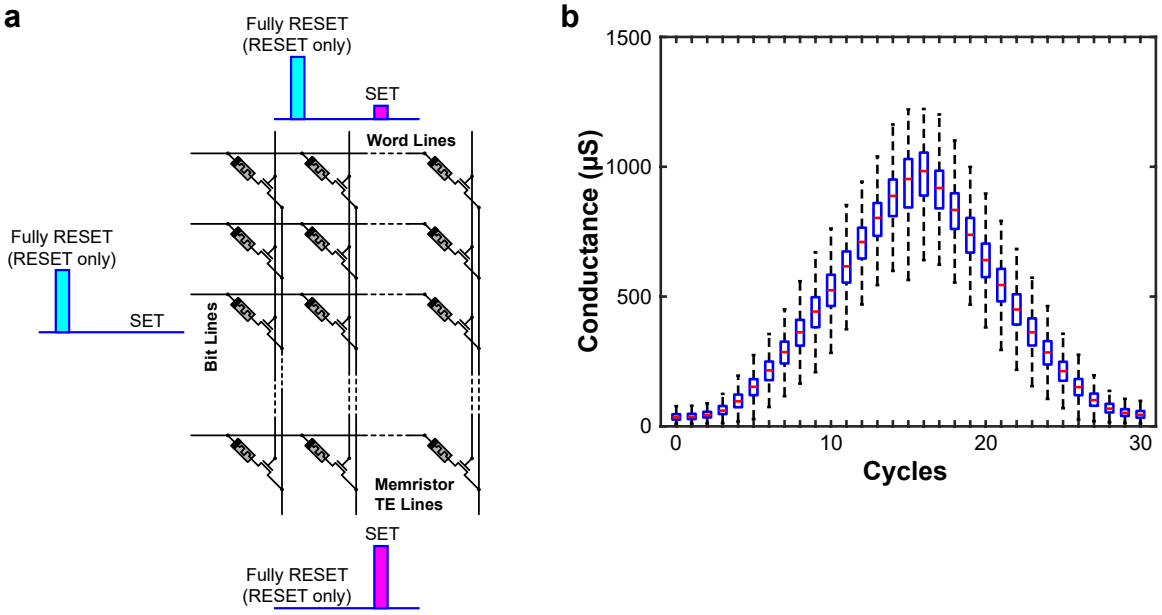
In situ training of feed-forward and recurrent convolutional memristor networks

Zhongrui Wang[1,8], Can Li[1,2,8], Peng Lin[1,8], Mingyi Rao[1], Yongyang Nie[1], Wenhao Song[1], Qinru Qiu[3], Yunning Li[1], Peng Yan[1], John Paul Strachan[2], Ning Ge[2], Nathan McDonald[4], Qing Wu[4], Miao Hu[5], Huaqiang Wu[6], R. Stanley Williams[7], Qiangfei Xia[1*] and J. Joshua Yang[1*]

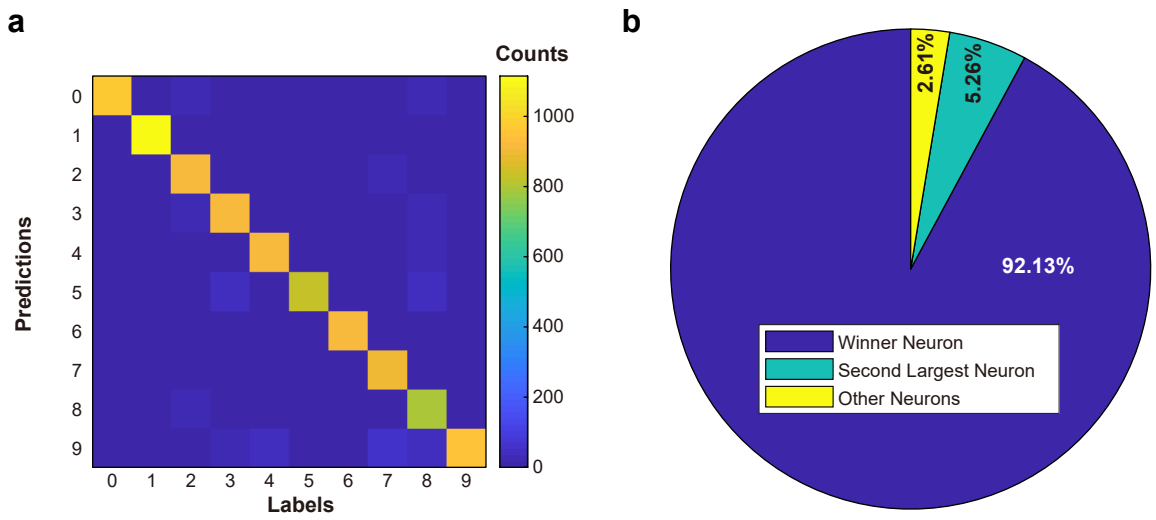
[1]Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA, USA. [2]Hewlett Packard Labs, Hewlett Packard Enterprise, Palo Alto, CA, USA. [3]Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY, USA. [4]Information Directorate, Air Force Research Laboratory, Rome, NY, USA. [5]Department of Electrical and Computer Engineering, Binghamton University, Binghamton, NY, USA. [6]Institute of Microelectronics, Tsinghua University, Beijing, China. [7]Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA. [8]These authors contributed equally: Zhongrui Wang, Can Li, Peng Lin. *e-mail: qxia@umass.edu; jjyang@umass.edu



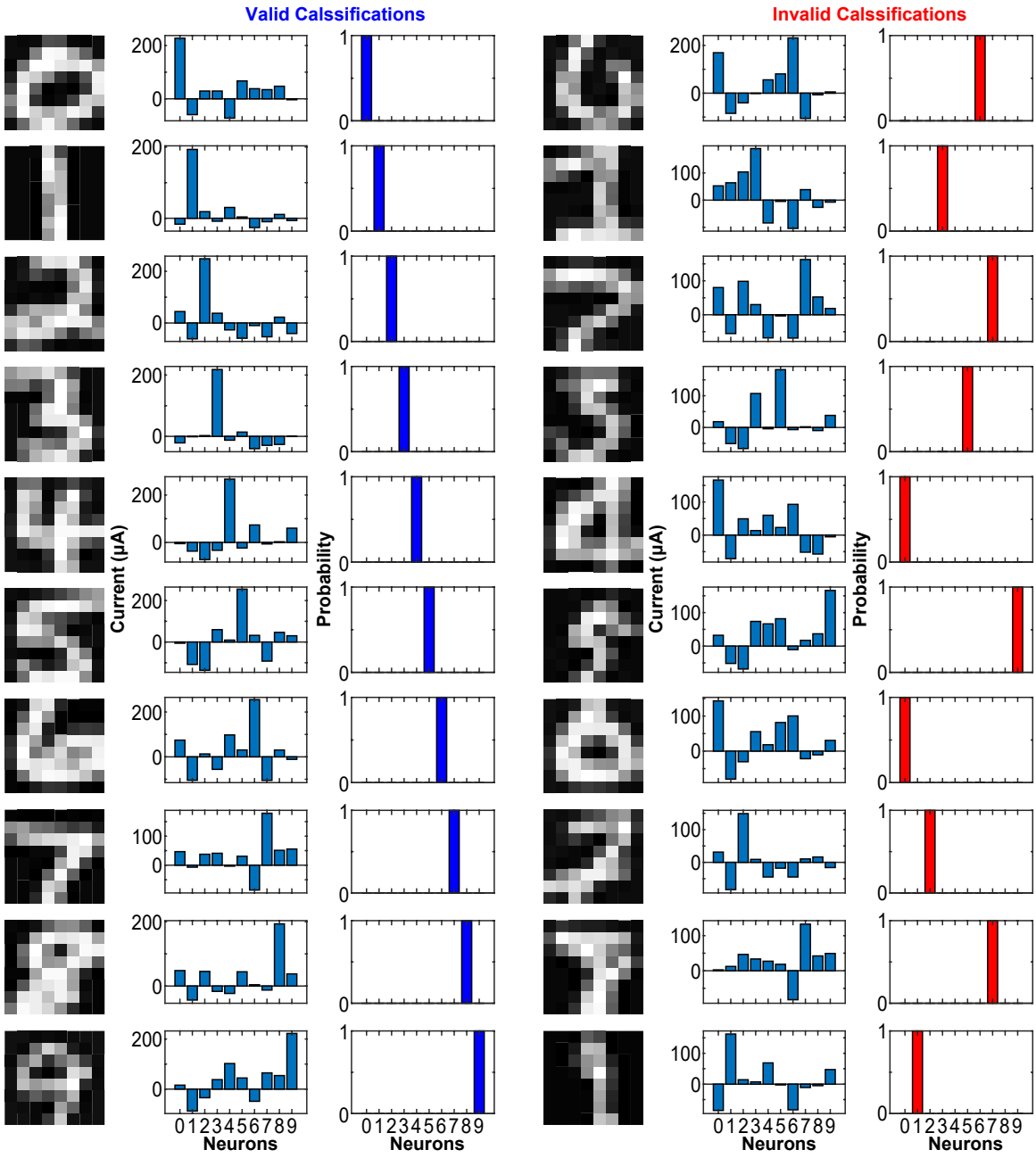
Supplementary Figure 1 | The hybrid analogue-digital computing system. The system consists of the 3 parts, the 1-transistor 1-memristor (1T1R) analogue memristor array, the printed circuit boards (PCBs, with on board analogue-to-digital converters or ADCs, digital-to-analogue converters or DACs, and trans-impedance amplifiers or TIAs), the general digital processor (microcontroller or MCU, and personal computer or PC). The 1T1R chip provides the hardware synapses and their associated topology. The PCBs implement part of the neuron functions. (The pre-synaptic DACs apply voltage signals to the bit lines of the 1T1R array, while the post-synaptic TIAs and ADCs read the summed currents across the memristor synapses of each column.) The PCBs also implement weight update and readout. The digital processors apply arbitrary activation functions to the current outputs of the post-synaptic modules, in addition to error backpropagation with physically acquired weights and gradients calculation with the RMSprop optimizer.



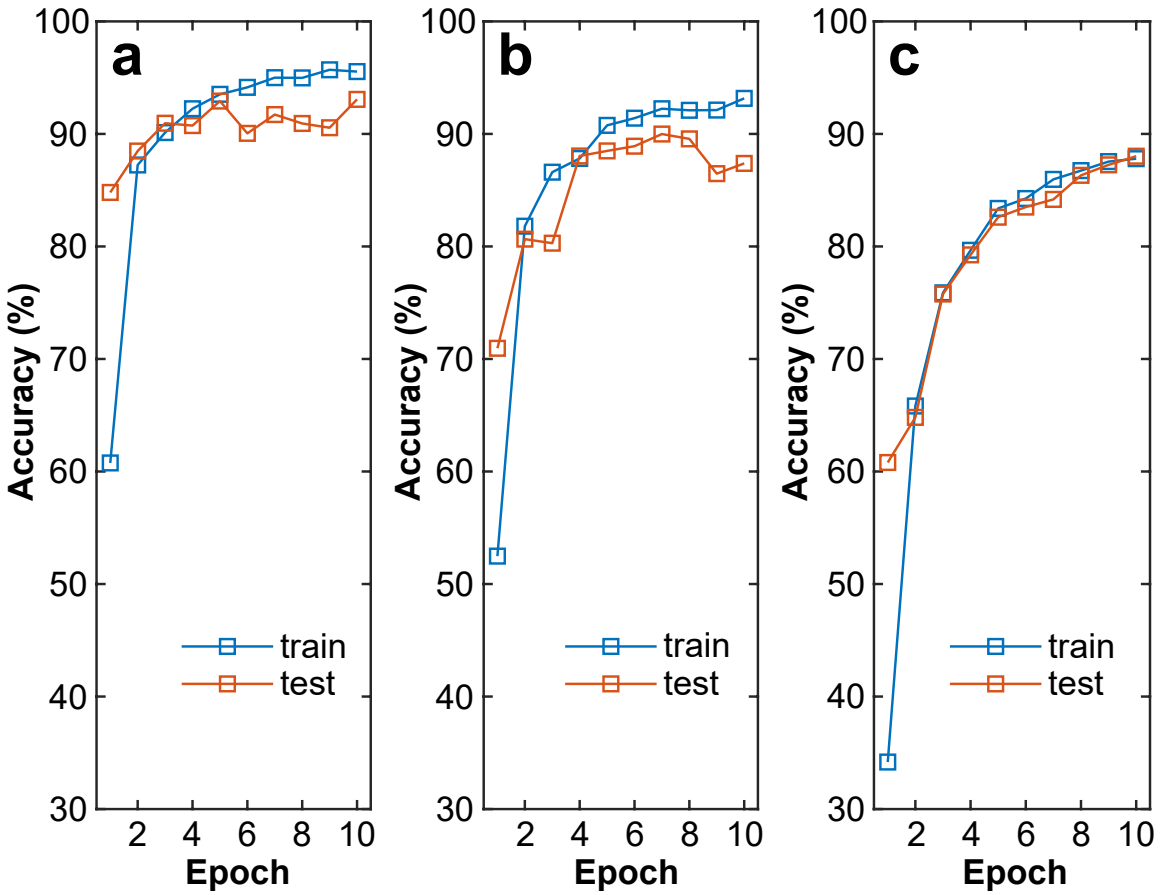
Supplementary Figure 2 | One-shot blind analogue programming of the 1T1R memristors. a, Scheme of the one-shot programming. Positive voltages were applied to the memristor top electrode (TE) lines and the word lines to SET the memristors. Positive voltages were applied to the bit lines and the word lines to fully RESET the memristors followed by a SET operation to achieve targeted conductance. (see Method and Table 1) **b**, Analogue SET and RESET with linearly varying word line voltages (gate voltages of the transistors). Statistical distribution of the conductance of the 128 × 64 memristor array after receiving 15 SET operations (All word line voltages were linearly ramped from 1 V to 2.4 V with a step 0.1 V in 15 cycles.) followed by 15 RESET operations (All word line voltages were linearly ramped down from 2.4 V to 1 V with a step 0.1 V in 15 cycles.), showing the linear and symmetric conductance tuning in one-shot without feedback. The red dots indicate the medians, and the bottom and top edges of the blue boxes indicate the 25th and 75th percentiles, respectively. The dashed lines extend to the most extreme data points. Notice the gain of the TIA limited the maximum readable conductance to ~1250 μS.



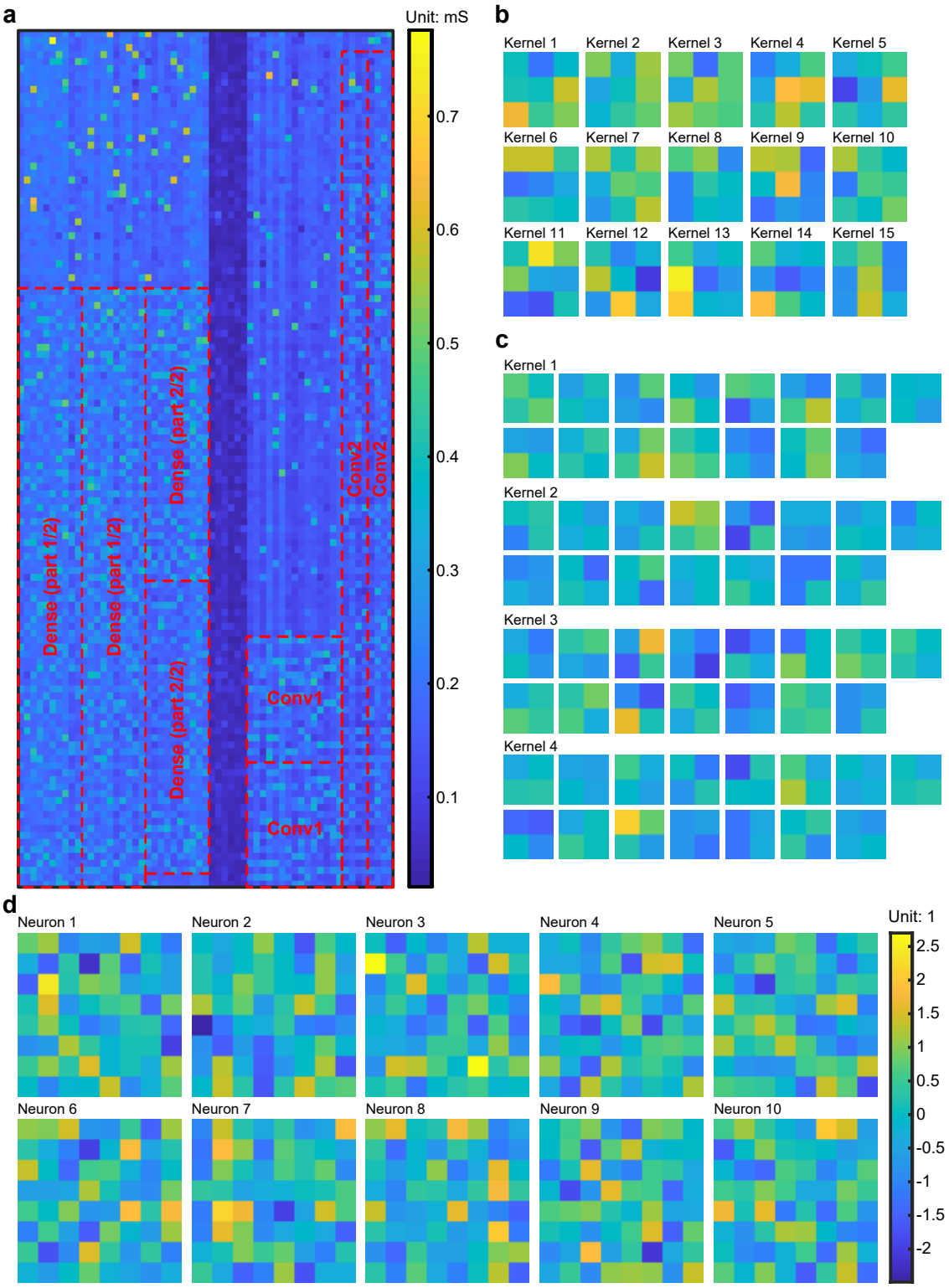
Supplementary Figure 3 | Inference statistics of the 5-level convolutional neural network (CNN). a, Graph illustrating the statistics of the winner neurons of different labels for all 10,000 MNIST test-set images. The most common misclassification was with the handwritten digital '7' which was likely to be identified as '9'. **b**, The pie chart of the classification, showing that the winner neurons have made correct predictions in 92.13% cases. The neurons of the second largest output matched with the labels in 5.26% cases.



Supplementary Figure 4 | Representative inference examples of the 5-level CNN. Examples of the valid classifications of the 10 digits (left 3 columns) and invalid classifications (right 3 columns). The second and fifth columns illustrate the raw output currents of the fully connected layer neurons. The corresponding Bayesian probabilities based on the softmax function are with the third and sixth columns. More data is shown in Supplementary Video 2.

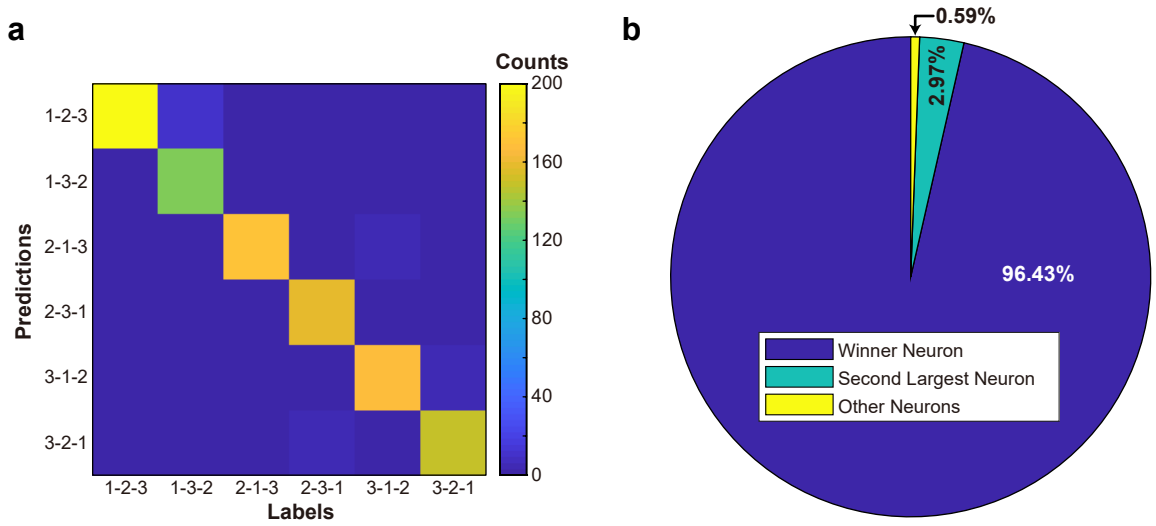


Supplementary Figure 5 | Simulated impact of memristor programming noise over the generalization error. a-c, Training the first 5,000 samples of MNIST with simulated 1T1R crossbars of programming error of 0.4 μS (the standard deviation of normal distributed noises), 4 μS, and 8 μS, respectively. It clearly shows the programming noise reduces the difference between the training accuracy (average of all in-batch training accuracy over the epoch) and the test accuracy.

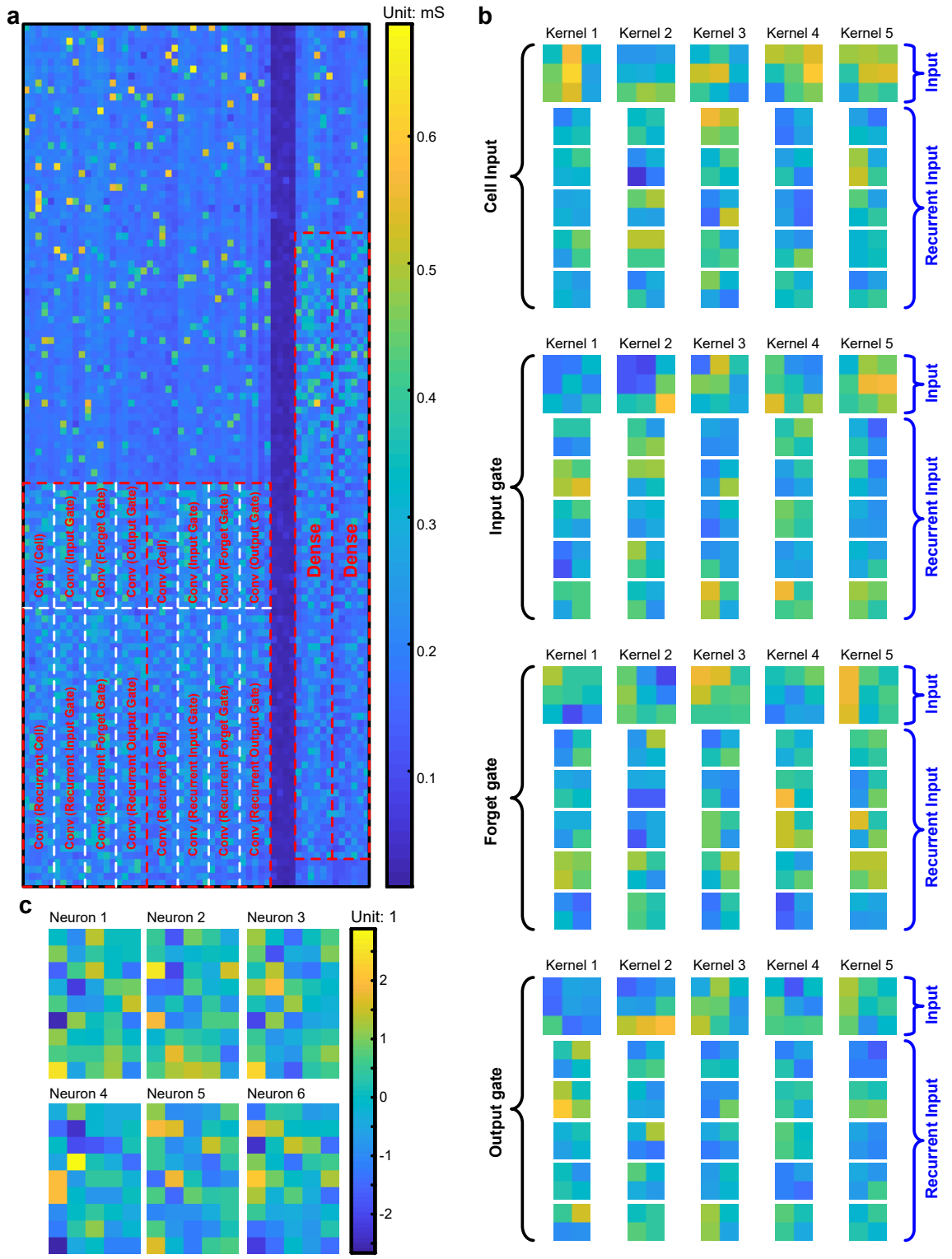


Supplementary Figure 6 | Post-training conductance map and weights of the 5-level CNN. a, The conductance of the 123 × 59 subarray used to implement the CNN after training on the 60,000 MNIST images for 2 epochs. **b**, The corresponding weights of the 15 kernels of size 3 × 3 of the first convolutional layer. Each weight is calculated by

dividing the averaged conductance differences of the 2 differential pairs by the constant conductance-to-weight ratio R_{gw} (see Method). **c**, The corresponding weights of the 4 kernels of size 2 × 2 (×15) of the second convolutional layer. Each graph is a 2-dimensional slice of the volumetric kernel along its depth. **d**, The corresponding weights of the 64 × 10 fully connected layer. Each graph of 8 × 8 pixels represents the 64 × 1 weight vector of a neuron after reshaping. The conductance and weights evolution during the *in situ* training is with Supplementary Video 1.

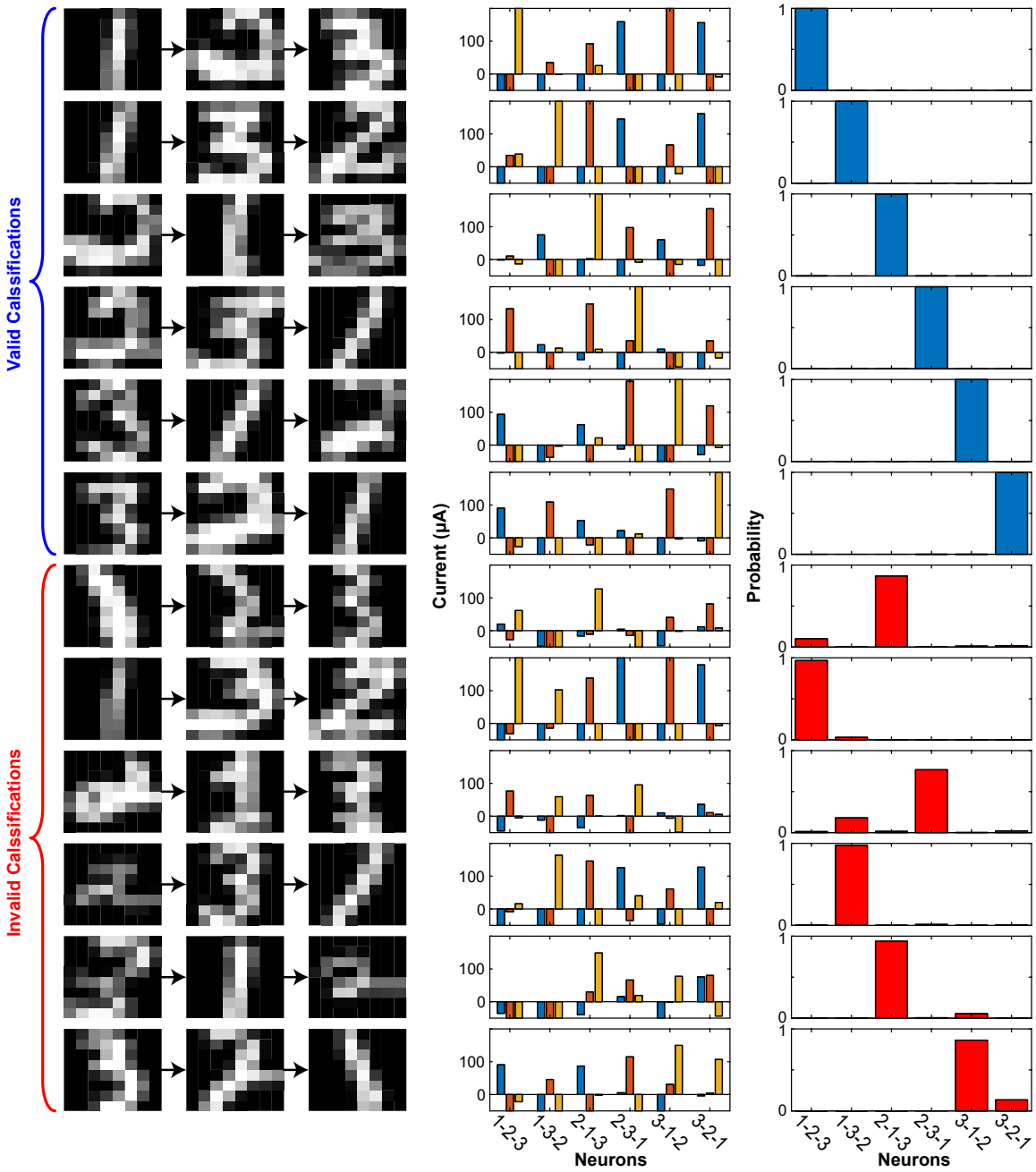


Supplementary Figure 7 | Inference statistics of the convolutional long-short term memory (ConvLSTM). a, Graph illustrating the statistics of the winner neurons of different labels for all 1,010 test-set MNIST-sequences. The most common misclassification was with the sequence '1-3-2' which was likely to be identified as '1-2-3'. **b**, The pie chart of the classification, showing that the winner neuron has made correct predictions in 96.43% cases. The neurons of the second largest output matched with the labels in 2.97% cases.

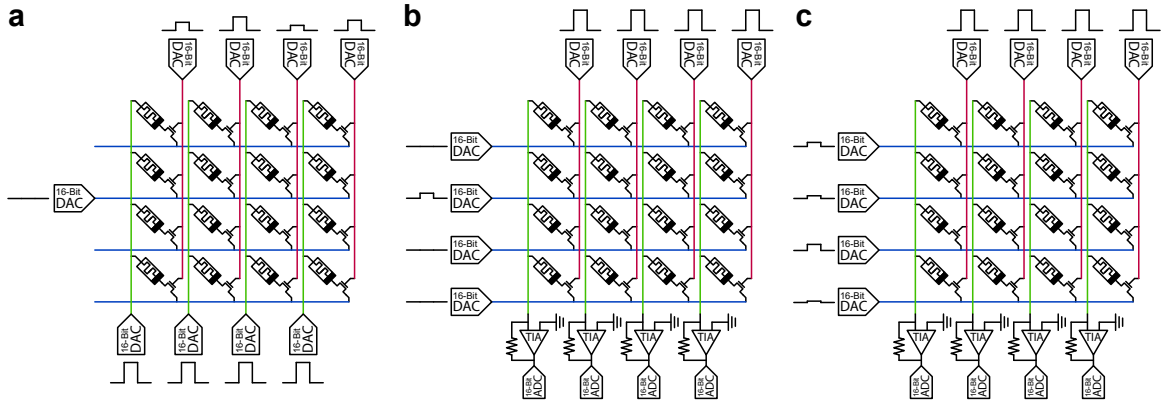


Supplementary Figure 8 | Post-training conductance map and weights of the ConvLSTM. a, The conductance of the 124 × 56 subarray used to implement the ConvLSTM after training on the 5,958 MNIST-sequences for 2 epochs. **b**, The corresponding weights of the 5 input kernels of size 3 × 3 and 5 recurrent input kernels of

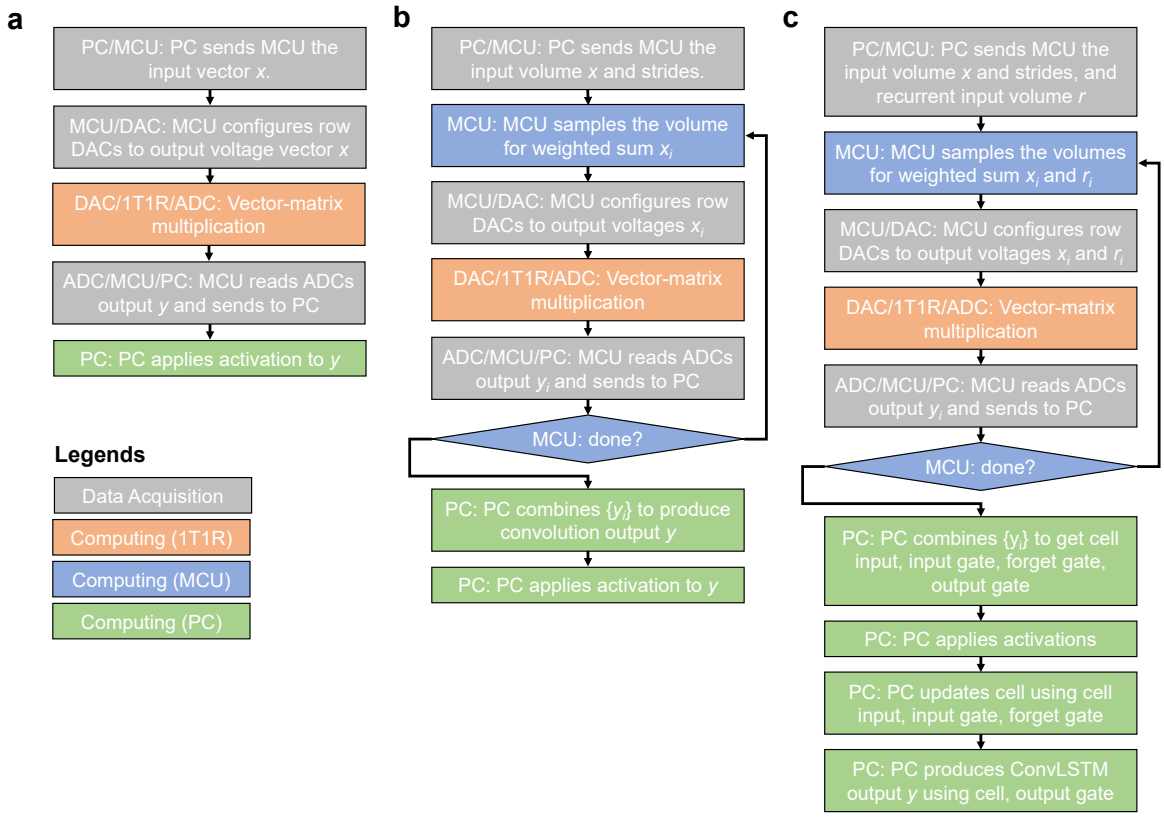
size 2 × 2 (× 5) of the cell input, input gate, forget gate, and output gate of the ConvLSTM layer. Each weight is calculated by dividing the averaged conductance differences of the 2 differential pairs by the constant conductance-to-weight ratio R_{gw} (see Method). **c**, The corresponding weights of the 45 × 6 fully connected layer. Each graph of 9 × 5 pixels represents the 45 × 1 weight vector of a neuron after reshaping. The conductance and weights evolution during the *in situ* training is with Supplementary Video 3.



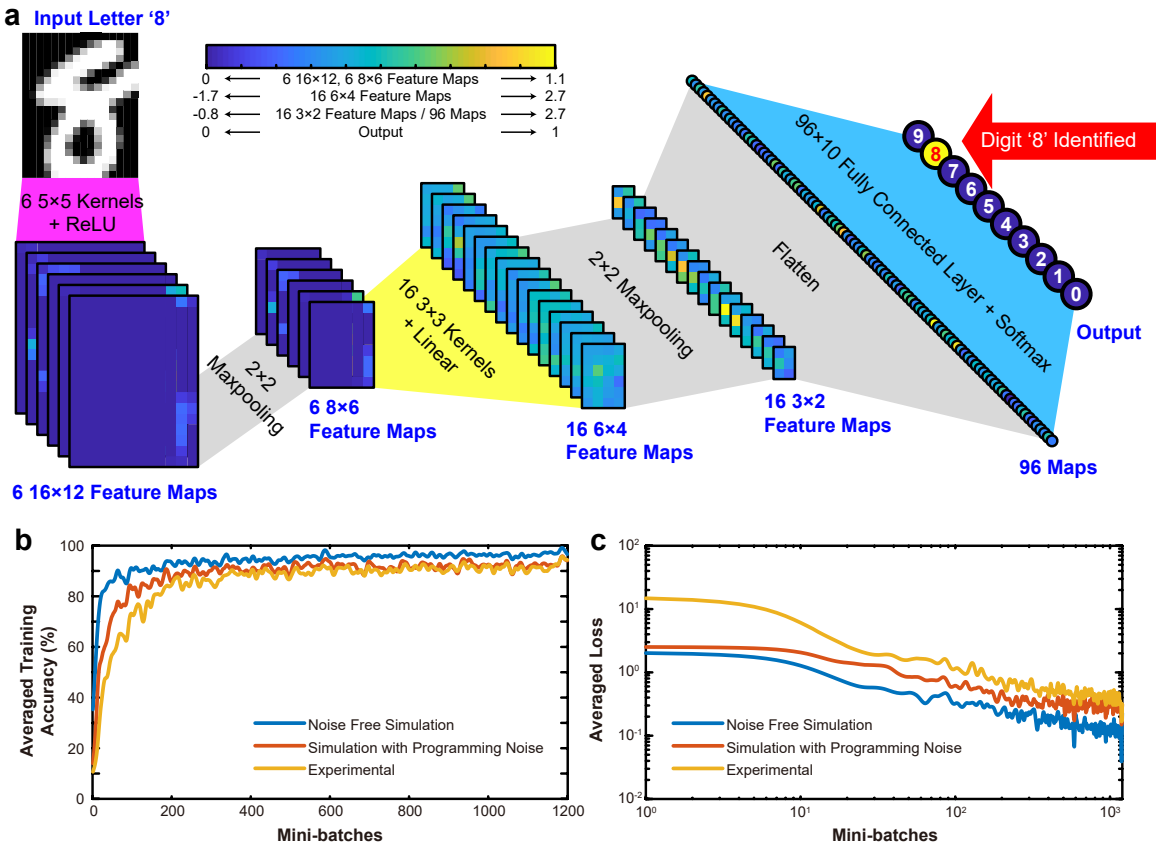
Supplementary Figure 9 | Representative inference examples of the ConvLSTM. Examples of the valid classifications of the 6 sequences (upper 6 rows) and invalid classifications (lower 6 rows). The middle column illustrates the raw output currents of the fully connected layer neurons at different time steps (time step 1: blue; time step 2: red, time step 3: orange). The corresponding Bayesian probabilities (of the last time step) based on the softmax function are with the right column. More data is shown in Supplementary Video 4.



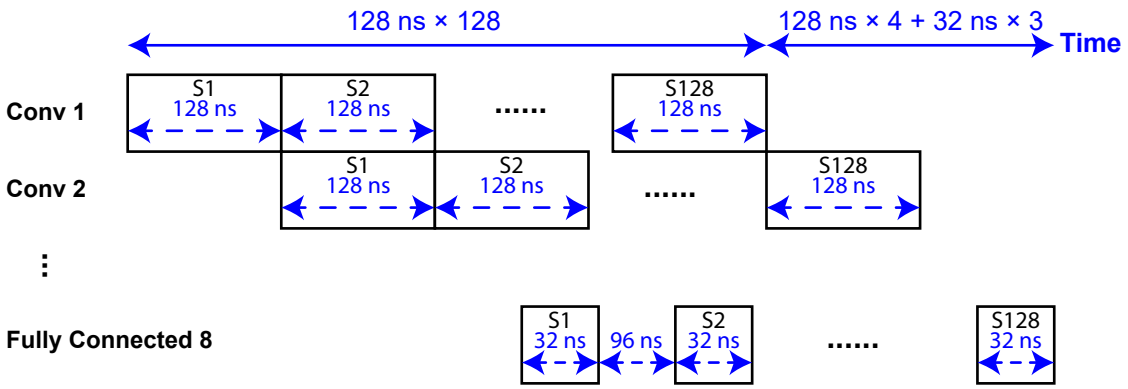
Supplementary Figure 10 | The analogue SET programming, conductance read-out, and vector-matrix multiplication schemes. a, The analogue SET programming flowchart. The 1T1R array was programmed row-by-row. For each row, the microcontroller set the DAC output voltages of all the memristor TE lines (green) to V_{SET} and word lines (red) to target conductance dependent V_{gate} . (see Method) Unselected bit lines (blue) were floated. **b**, The conductance readout flowchart. The conductance of the array was read row-by-row. In each time step, the selected bit line was biased to V_{Read} while the rest bit lines were grounded. The microcontroller read the ADCs of all memristor TE lines in each time step. All word lines were set to V_{gate} = 5 V in all time steps. **c**, The vector-matrix multiplication flowchart. The bit line DACs were configured to produce the input vector $\vec{V}$. The microcontroller read the ADCs of all memristor TE lines. All word lines were set to V_{gate} = 5 V.



Supplementary Figure 11 | The block diagrams of the forward pass of **a**, fully connected layer, **b**, 2-dimensional convolution layer, **c**, ConvLSTM layer. For the fully connected layer, the vector-matrix multiplication and activations are performed by the 1T1R crossbar and the PC, respectively. For the convolution, low level computation (in proximity to the 1T1R crossbar) by MCU samples the volume for weighted sum with kernels. For the ConvLSTM layer, additional PC computation updates the cell status and produces the ConvLSTM output.



Supplementary Figure 12 | Architecture and experimental performance of a LeNet-5-like CNN. a, The exemplary forward pass in the inference. The 8-bit grayscale input of size 20 × 16 (cropped from original MNIST without down-sampling) was convolved by the 6 memristor kernels of size 3 × 3 with rectified linear unit (ReLU) activation followed by the max pooling of size/stride 2 × 2. The feature maps are then convolved by sixteen 3 × 3 kernels, followed by the second 2 × 2 max pooling layer, before being fed to the fully connected 10-way softmax output layer. **b**, The smoothed experimental in-batch accuracy (loss) increased (decreased) over the course of *in situ* training. The experimental curves are indistinguishable from the simulation considering the programming noise, tightly following the noise-free simulation.



Supplementary Figure 13 | Schematic illustration of the time span of the minibatch forward pass with the hybrid analogue-digital system for AlexNet[1]. S1, S2, and S128 refers the first, second, and 128th sample of the minibatch. The forward passes of different samples are "parallel" because of the sequential structure of the AlexNet.

Supplementary Table 1 | Summary of the network architecture, the number of operations per cycle, the number of cycles per sample in the forward pass of the CNN in Fig. 1-2 of the main text.

Layer	Dimension	Output Dimension	No. operations per cycle	No. cycles per sample	No. operations per sample
Input layer 0		8×8			
Convolution layer 1	(3×3×1) ×15	8×8×15	270	64	17,280
Convolution layer 2	(2×2×15) ×4	8×8×4	480	64	30,720
Max pooling layer 3	2×2	4×4×4			
Fully connected layer 4	64×10	10	1280	1	1,280

Supplementary Table 2 | Summary of the network architecture, the number of operations per cycle, the number of cycles per sample in the forward pass of the ConvLSTM network in Fig. 3-4 of the main text.

Layer	**Dimension**	**Output Dimension**	**No. operations per cycle**	**No. cycles per sample**	**No. operations per sample**
Input layer 0		8×8			
Convolutional LSTM layer 1	(3×3×1) ×20 for input (2×2×5) ×20 for recurrent input	6×6×5	1,160	36×3=108 (3 time steps)	125,280
Max pooling layer 2	2×2	3×3×5			
Fully connected layer 3	45×6	6	540	1	540

Supplementary Table 3 | Performance comparison of our CNN (Fig. 1-2) and a LeNet-5-like CNN. Both CNNs show similar accuracy with different training environments, indicating a larger ratio between the accuracy and the number of weights of our CNN. The accuracy of different environments also reveals the two key factors that lower down the accuracy in the experimental runs, the bounded weights and programming noise. (*The inputs are linearly remapped to [-0.2, 0.2] in the experimental run.).

Training Environments	**Our CNN (1,015 weights)**		**LeNet-5 like CNN (1,974 weights)**	
	Epoch 1 Accuracy	**Epoch 2 Accuracy**	**Epoch 1 Accuracy**	**Epoch 2 Accuracy**
Google Keras	95.75%	96.30%	95.75%	97.11%
Our customized framework with software backend	95.52%	96.58%	96.05%	96.82%
Simu-array (0.04 μS programming noise)	92.53%	94.57%	94.63%	95.55%
Simu-array (8 μS programming noise)	91.06%	89.23%	91.29%	89.56%
Experimental run	86.99%*	92.13%*	90.44%	90.84%

Supplementary Table 4 | The impact of down-sampling MNIST dataset on the training accuracy with fixed dimension of convolution kernels and fully connected layers. The original MNIST dataset of size 28 × 28 is first cropped to 20 × 20 by keeping the central region. Bicubic down-sampling is then applied to produce 16 × 16, 12 × 12, and 8 × 8 datasets. The results of Google Keras, our customized framework of software backend, and the simulated memristor array with 8 μS programming noise show no clear down-sampling impact on the accuracy.

Input size	Network architecture					Google Keras		Software		Simulated memristors	
	Layer 1: Conv 2D	Layer 2: Conv 2D	Layer 3: Max Pooling	Layer 4: FC	Total Para	Epoch 1 Accuracy	Epoch 2 Accuracy	Epoch 1 Accuracy	Epoch 2 Accuracy	Epoch 1 Accuracy	Epoch 2 Accuracy
20×20	3×3 (×15)	2×2 (×4)	5×5	64×10	1,015	96.31%	97.01%	94.04%	95.10%	90.78%	91.32%
16×16			4×4			96.38%	97.26%	95.67%	96.35%	92.23%	90.85%
12×12			3×3			96.99%	96.42%	94.80%	96.57%	90.39%	92.16%
8×8			2×2			95.75%	96.30%	95.52%	96.58%	91.06%	89.23%

Supplementary Table 5 | The impact of down-sampling MNIST dataset on the training accuracy with fixed dimension of convolutional kernels and max pooling layers. The original MNIST dataset of size 28 × 28 is first cropped to 20 × 20 by keeping the central region. Bicubic down-sampling is then applied to produce 14 × 14 and 8 × 8 datasets. The relatively large accuracy change of the simulated array with programming noise is attributed to the change of the total number of trainable parameters that could compensate the inaccuracy of the programming.

Input size	Network architecture					Google Keras		Software		Simulated memristors	
	Layer 1: Conv 2D	Layer 2: Conv 2D	Layer 3: Max Pooling 2D	Layer 4: FC	Total Para	Epoch 1 Accuracy	Epoch 2 Accuracy	Epoch 1 Accuracy	Epoch 2 Accuracy	Epoch 1 Accuracy	Epoch 2 Accuracy
20×20	3×3 (×15)	2×2 (×4)	2×2	400×10	4,375	97.28%	97.67%	96.66%	97.39%	93.39%	93.63%
14×14				196×10	2,335	97.46%	97.85%	96.58%	97.13%	92.23%	92.76%
8×8				64×10	1,015	95.75%	96.30%	95.52%	96.58%	91.06%	89.23%

Supplementary Table 6 | Summary of the implementation of AlexNet[1] on a projected hybrid analogue-digital system. The kernel dimension for the convolution layers (Conv2d) are 96 11×11×3 kernels, 256 5×5×96 kernels, 384 3×3×256 kernels, 384 3×3×384 kernels, and 256 3×3×384 kernels, respectively. Each kernel is mapped to a column of the 1T1R crossbar with a bias (e.g. For Conv2D 1, the 1T1R array size is [11×11×3+1, 96].) The output feature maps of the convolution layers are 55×55×96, 27×27×256, 13×13×384, 13×13×384, and 13×13×256, respectively. The number of strides (different volumes for weighted sum) is thus 55×55, 27×27, 13×13, 13×13, and 13×13 respectively. Notice more hardware copies of the convolution kernels helps to achieve a balance between the throughput and the area-energy efficiency. The dimension of the fully connected layers (FC) are 9216×4096, 4096×4096, and 4096×1000, respectively.

Layer	**Single 1T1R array size (2D)**	**Strides**	**Array Copies**	**No. of 1T1Rs**	**No. of ADCs**	**No. of Analogue Operations**	**Time (ns)**
Conv2D 1	[364,96]	3,025	757	26,452,608	4,542	211,411,200	128
Conv2D 2	[2,401,256]	729	183	112,482,048	2,928	896,168,448	128
Conv2D 3	[2,305,384]	169	43	38,060,160	1,032	299,170,560	128
Conv2D 4	[3,457,384]	169	43	57,081,984	1,032	448,690,944	128
Conv2D 5	[3,457,256]	169	43	38,054,656	688	299,127,296	128
FC 6	[9,216,4,096]	1	1	37,748,736	256	75,497,472	32
FC 7	[4,096,4,096]	1	1	16,777,216	256	33,554,432	32
FC 8	[4,096,1,000]	1	1	4,096,000	63	8,192,000	32
	Total parameters n_{para}= 62,369,152			Total 1T1Rs n_{1T1R}= 330,753,408	Total ADCs n_{ADC}= 10,797	Total operation n_{OP} = 2,271,812,352	

Supplementary Note 1: Comparison with another LeNet-5-like CNN

We compare the simple CNN of Fig. 1-2 with a LeNet-5-like network. The LeNet-5-like network differs from the original LeNet-5 by using a single fully connected layer and different activations, which allows it to be implemented on the same 128 × 64 1T1R crossbar array. (See Supplementary Figure 12a) The input to the LeNet-5-like network are 20 × 16 pixels cropped from original MNIST dataset without down-sampling. The inputs are convolved by six 5 × 5 kernels, followed by the first max pooling layer of size/stride 2 × 2. The feature maps are then convolved by sixteen 3 × 3 kernels, followed by the second 2 × 2 max pooling layer, before being fed to the fully connected 10-way softmax output layer. Like the training in Fig. 2, the experimental curve in Supplementary Figure 12b follows a consistent trend with the noise free simulation. It is also nearly indistinguishable from the simulated curve with the assumed programming noise (standard deviation of 10 μS).

Supplementary Table 3 compares the performance of the two networks. It shall be noted that our CNN could take cropped MNIST inputs without down-sampling by using a larger max pooling window/stride. Such change will not decrease the accuracy but rather even slightly increase it. (see Supplementary Note 2 for the MNIST down-sampling impact). Both our CNN and the LeNet-5 like CNN show similar accuracy in Google Keras, our framework with software backend, simulation with programming noise, as well as experimental runs. Given our CNN uses 1,015 weights while the LeNet-5-like CNN uses 1,974 weights, our CNN architecture thus shows a better ratio between the accuracy and the number of weights, which better illustrates the advantage of weight sharing.

Supplementary Table 3 also reveals the two key factors that lower down the accuracy in the experimental runs, which are bounded weights and programming noise. Since we use a finite range of memristor conductance and a constant factor to convert conductance to weight (see Table 1), therefore, the span of each weight is numerically bounded. This possibly leads to the observed accuracy decrement in the simulated array with 0.04 μS programming noise. On the other hand, the memristor differential pair is equivalent to a digital weight of ~3.7 bit (under 1σ uncertainty), which is one of the best analogue properties for emerging memories with a single blind weight update[2]. The precision of 1T1R synapses could be further improved with material engineering (e.g. epitaxial growth of the dielectric memristor material stack[3]) and synapses consisting of multiple 1T1Rs[4].

Supplementary Note 2: The MNIST dataset down-sampling impact to the performance of the CNN

We simulate the down-sampling influence to the performance of the CNN of Fig. 1-2 with identical dimension of convolution kernels and fully connected layers. Since the input size varies, therefore different window sizes and strides are used for the max pooling layer. As shown in Supplementary Table 4, the down-sampling of the input shows no clear effect on the accuracy with different implementation environments (Google Keras, our customized framework using software backend, and the simulated memristor arrays with 8 μS programming noise).

Alternatively, we also investigate the cases where the shape of the convolution kernels and max pooling layers are fixed, while varying the size of the fully connected layers, as summarized in Supplementary Table 5. The three implementation environments give 1.37%, 0.81%, and 4.4% accuracy difference (second epoch) between the 20×20 and 8×8 datasets. The relatively large accuracy change of the simulated array with 8 μS standard distribution of programming noise is attributed to the change of the total number of trainable parameters that could compensate the inaccuracy of programming.

Supplementary Note 3: The projected system-level area and energy efficiency of the 128 × 64 1T1R memristor crossbar

In a system centric view, a hybrid analogue-digital peripheral circuit may further benefit overall area and energy efficiency with the currently available technologies, although our proof-of-concept measurement system employed the fully analogue approach. Various signal representations (analogue voltages[5,6], analogue pulse width modulation[7-11], and digital[12-14]) and the corresponding signal generation/acquisition circuits are still under studies for memristor crossbars. The optimal signal representation should be application-specific and depending on the available technology. For instance, the high-resolution DACs could be replaced by pulse width modulators for the representation of the analogue signals.[15] Such digital signal representation could also work with larger resistance non-Ohmic memristors, at the expense of speed overhead compared to the fully analogue approach.

With the digital signal representation, the output electric currents of the memristor crossbar are integrated by integrator amplifiers, and then sampled and hold by a sequential readout with the analogue-digital converters (ADCs) shared among output terminals. The number of shared ADCs is determined by the trade-off between the baud rate of input signals, latency of the crossbar array / integrator amplifier, and the sampling rate of the ADC. A state-of-art 6-bit (which is good enough for most of the neural network scenario) ADC design of 40 nm technology node could achieve 1G samples/s while consuming 1.26 mW energy[16]. Assume the settling time for the memristor crossbar (with 64 column outputs) and the integrator amplifier is 16 ns, the same amount of time to represent 6-bit input data at a 2G baud rate. Then each ADC samples 16 columns would achieve the maximum throughput by enabling the pipelined operation. Alternatively, the digitization of the analogue output currents could be implemented on energy-area efficient current sense amplifiers with input-dependent reference currents[13,14].

Mathematically, for the $N \times M$ sized 1T1R memristor crossbar, the throughput is $T_{\text{chip}} = \frac{N \times M \times 2}{t_{\text{period}}}$, since the array performs $N \times M$ additions and the same number of multiplications per analog operation. Given $N = 128$, $M = 64$, $t_{\text{period}} = 32\,\text{ns}$, the throughput is $T_{\text{chip}} = 5.12 \times 10^{11}$ OPS for 6-bit operations (the memristor used here could show 6-bit or 64 levels[17]).

The chip area is mainly determined by the size of the 1T1R cells ($A_{1\text{T1R}}$) and the ADCs ($A_{\text{ADC}} = 0.00058\,\text{mm}^2$)[18], which could be estimated as $A_{\text{chip}} = N \times M \times A_{1\text{T1R}} + \left\lceil \frac{M}{16} \right\rceil \times A_{\text{ADC}}$. Assume the each 1T1R cell is of a footprint $100F^2$ (where $F = 40\,\text{nm}$, the technology node of the ADC), the chip area is estimated to be $A_{\text{chip}} = 0.0036\,\text{mm}^2$.

For power consumption, the major portion would be power of the 1T1R cell ($P_{1\text{T1R}}$) and the ADC ($P_{\text{ADC}} = 1.26\,\text{mW}$), and others are negligible according to our previous work[18], which is $P_{\text{chip}} = 0.5 \times \left(N \times M \times P_{1\text{T1R}} + \left\lceil \frac{M}{16} \right\rceil \times P_{\text{ADC}} \right)$ as 1T1R cells and ADCs only operate in half of the t_{period}. Assuming an averaged memristor biasing voltage 0.1V (half of the peak biasing voltage 0.2V), the cell power is $P_{1\text{T1R}} = \frac{0.1\text{V}^2}{10^4 \Omega} = 10^{-6} W$ and the system power is $P_{\text{chip}} = 6.62\,\text{mW}$, which could be further improved by employing lower conductance memristor cells.

This simple estimation indicates an area efficiency $\frac{T_{\text{chip}}}{A_{\text{chip}}} \cong 141\ \text{TOPS/mm}^2$, and an energy efficiency $\frac{T_{\text{chip}}}{P_{\text{chip}}} \cong 77.4\ \text{TOPS/W}$ for 6-bit operations. Furthermore, the memristor-based processing-in-memory computing system could get rid of the off-chip dynamic random-access memory (DRAM) because the memristor crossbar serves dual roles, the main memory and the multiply accumulator. On the contrary, most ASIC accelerators rely on off-chip DRAM chips to host the parameters of the neural network, which incurs extra area and power consumption.

Supplementary Note 4: The projected system-level area and energy efficiency of a large-scale 1T1R-based computing system

Here we present a simple system performance estimation of a large-scale hybrid analogue-digital computing system using 1T1Rs. The memristor-based computing system could feature advantages in both area and energy efficiencies over conventional digital systems (see Supplementary Note 3). This is not only valid for the small-scale proof-of-concept problem studied here, but it can also be applied to a real-world problem, for instance the AlexNet[1] while retaining the advantages on efficiencies, which has also been revealed by similar proposed memristor-based platforms such as Prime[19], ISAAC[18], Atomlayer[20], and PUMA[20].

Supplementary Table 6 summarizes the proposed implementation of the AlexNet on a memristor-based hybrid analogue-digital computing system. Please note that typical ASIC systems need off-chip DRAM to store the $n_{\text{para}} = 62{,}369{,}152$ parameters of the AlexNet. Assume each parameter is 6-bit (same with the representation capability of our memristors[17]), $n_{\text{para}} \times 6 = 374{,}214{,}912$ DRAM cells are needed. On the other hand, we propose to use less 1T1R cells $n_{1\text{T1R}} = 330{,}753{,}408$ with the hybrid analogue-digital system. Since DRAM and 1T1R cells share identical capacitor structures, so the footprint of the DRAM cell is roughly identical with that of the 1T1R cell. As each 1T1R could in principle represent 6-bit[17], more copies of the hardware kernels could be deployed for specific layers, which helps to achieve a balance between the throughput and the area-energy efficiency. (See Supplementary Table 6)

To compute the throughput, we first estimate the average forward pass time per sample. Please note the forward pass could be "pipelined" because of the sequential architecture of AlexNet. (Supplementary Figure 13) Therefore, the average forward pass time per sample is $t_{\text{sample}} = \max_{1 \le i \le 8} t_i + \frac{\sum_{j \ne i} t_j}{n_{\text{batch}}}$, where t_j refers to the forward pass time of i-th layer. Given the minibatch size $n_{\text{batch}} = 128$ for AlexNet[1], $t_{\text{sample}} = 132.75$ ns. The throughput of the memristor-based system is thus $T_{\text{chip}} = \frac{n_{\text{OP}}}{t_{\text{sample}}} = \frac{2{,}271{,}812{,}352}{132.75\text{ ns}} \cong 17.1$ POPS. Note extra time and energy could be incurred by the inter-array communication fabric, but usually less dominating compared to those of in-array signal generation/acquisition[18-21].

In terms of the area efficiency, since 1T1R cells and DRAM cells are of the same structure, we assume they have the same footprint $A_{1\text{T1R}} = A_{\text{DRAM}}$. For the memristor-based system, the 10,797 ADCs[16] cost $n_{\text{ADC}} \times A_{\text{ADC}} \cong 6.26\text{ mm}^2$ ($A_{\text{ADC}} = 0.00058\text{ mm}^2$, see Supplementary Note 3). If the overall area-efficiency is defined as the ratio between the throughput and the total chip area (including the DRAM), then the memristor-based system has the area efficiency $\frac{T_{\text{chip}}}{n_{1\text{T1R}} \times A_{1\text{T1R}} + n_{\text{ADC}} \times A_{\text{ADC}}} \cong 289\text{ TOPS/mm}^2$ ($A_{1\text{T1R}} = 100F^2$ where $F = 40$ nm, see Supplementary Note 3).

In terms of the energy efficiency, note that the total power (mean value per sample) of the system is $\frac{1}{t_{\text{sample}}} \times \sum_{i=1}^{8} \left[0.5 \times \left(P_{1\text{T1R}} \times n_{1\text{T1R},i} + P_{\text{ADC}} \times n_{\text{ADC},i}\right) \times t_i\right] \cong 145\text{ W}$ where $n_{1\text{T1R},i}$ and $n_{\text{ADC},i}$ are the number of 1T1Rs and ADCs of the i-th layer (The factor 0.5 is due to the fact that 1T1R cells and ADCs only operate in half of the t_{period}, see Supplementary Note 3). Therefore, the power/energy efficiency of the memristor-based

system is $\frac{T_{\text{chip}}}{\frac{1}{t_{\text{sample}}}\times\sum_{i=1}^{8}\left[0.5\times\left(P_{\text{1T1R}}\times n_{\text{1T1R},i}+P_{\text{ADC}}\times n_{\text{ADC},i}\right)\times t_i\right]} \cong 118\ \text{TOPS/W}$, for 6-bit operations.

In addition to forward pass, the memristor-based hybrid analogue-digital system also benefits the weight update in terms of energy and time. This is because the off-chip DRAM writing energy is relatively large, ~1.3 nJ per byte[22] (or interpolated write energy $E_{\text{DRAM}} \cong 163$ pJ per bit), and the DRAM bandwidth is usually small (e.g. the DRAM bandwidth of Google tensor processing unit[23] (TPU) generation 1 is $B_{\text{TPU}} = 34\ \text{GByte}\cdot\text{s}^{-1} \cong 2.92\times10^{11}\ \text{bit}\cdot\text{s}^{-1}$). On the other hand, 1T1R cells of the crossbar could be programmed column by column (or row-by-row). Therefore, the time overhead is the array of the largest number of columns, which is $\max_{1\le i\le 8} n_{\text{col},i} = 4{,}096$ here. Since each memristor could be completely programmed by a $t_{\text{program}} = 5$ ns pulse at 2.2 V,[24] the total programming time could be around $\left(\max_{1\le i\le 8} n_{\text{col},i}\right)\times t_{\text{program}} \cong 20.5$ μs, significantly smaller than the time to update all parameters of AlexNet in a Google TPU ($n_{\text{para}}\times 6/B_{\text{TPU}} \cong 1.28$ ms. Only memory bandwidth lag is counted while DRAM writing time is ignored). In addition, for memristor-based system, the maximum energy for programming a single 1T1R cell is $E_{\text{program}} = \frac{2.2\text{V}^2}{10\text{k}\Omega}\times t_{\text{program}} = 2.42$ pJ. Thus the maximum total programming energy to update all parameters of AlexNet is $E_{\text{progrm}}\times n_{\text{1T1R}} \cong 800$ μJ, which is also significantly smaller than the energy to update all parameters in DRAM with a digital system ($E_{\text{DRAM}}\times n_{\text{para}}\times 6 \cong 60.8$ mJ). The rest training processes, including the calculation of errors and gradients of trainable parameters, are implemented on conventional transistor-based digital logics.

In summary, the memristors not only replace DRAM for storing trainable parameters but also perform vector-matrix multiplications at where the data are stored, which breaks the von Neumann bottleneck. On the contrary, the throughput of transistor-based neural network accelerators is not scaling with the size of the network (or the number of trainable parameters, the size of the memory), while the energy efficiency is constrained by the off-chip DRAM accessing. Thus, our finding reveals that the real-world large-scale neural networks could be significantly benefited by the memristor-based hybrid analogue-digital system in terms of both area and energy efficiencies.

Supplementary References

1 Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. in *Advances in neural information processing systems.* 1097-1105 (2012).

2 Kim, G. H. *et al.* Four-Bits-Per-Cell Operation in an HfO_2-Based Resistive Switching Device. *Small* **13**, 1701781, (2017).

3 Choi, S. *et al.* SiGe epitaxial memory for neuromorphic computing with reproducible high performance based on engineered dislocations. *Nat. Mater.* **17**, 335-340, (2018).

4 Boybat, I. *et al.* Neuromorphic computing with multi-memristive synapses. *Nat. Commun.* **9**, 2514, (2018).

5 Prezioso, M. *et al.* Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* **521**, 61-64, (2015).

6 Bayat, F. M. *et al.* Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits. *Nat. Commun.* **9**, 2331, (2018).

7 Sheridan, P. M. *et al.* Sparse coding with memristor networks. *Nat. Nanotechnol.* **12**, 784-789, (2017).

8 Yao, P. *et al.* Face classification using electronic synapses. *Nat. Commun.* **8**, 15199, (2017).

9 Lin, Y. *et al.* Demonstration of Generative Adversarial Network by Intrinsic Random Noises of Analog RRAM Devices. in *2018 IEEE International Electron Devices Meeting (IEDM).* 3.4. 1-3.4. 4 (IEEE, 2018).

10 Zhou, Y. *et al.* Associative Memory for Image Recovery with a High‐Performance Memristor Array. *Adv. Funct. Mater.*, 1900155, (2019).

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

12 Mochida, R. *et al.* A 4M synapses integrated analog ReRAM based 66.5 TOPS/W neural-network processor with cell current controlled writing and flexible network architecture. in *2018 IEEE Symposium on VLSI Technology.* 175-176 (IEEE, 2018).

13 Chen, W.-H. *et al.* A 65nm 1Mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors. in *Solid-State Circuits Conference-(ISSCC), 2018 IEEE International.* 494-496 (IEEE, 2018).

14 Xue, C.-X. *et al.* A 1Mb Multibit ReRAM Computing-In-Memory Macro with 14.6 ns Parallel MAC Computing Time for CNN Based AI Edge Processors. in *2019 IEEE International Solid-State Circuits Conference-(ISSCC).* 388-390 (IEEE, 2019).

15 Jiang, H. *et al.* Pulse-Width Modulation based Dot-Product Engine for Neuromorphic Computing System using Memristor Crossbar Array. in *2018 IEEE International Symposium on Circuits and Systems (ISCAS).* 1-4 (IEEE, 2018).

16 Choo, K. D., Bell, J. & Flynn, M. P. Area-efficient 1GS/s 6b SAR ADC with charge-injection-cell-based DAC. in *Solid-State Circuits Conference (ISSCC), 2016 IEEE International.* 460-461 (IEEE, 2016).

17 Li, C. *et al.* Analogue signal and image processing with large memristor crossbars. *Nat. Electron.* **1**, 52-59, (2018).

18 Shafiee, A. *et al.* ISAAC: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *SIGARCH Comput. Archit. News* **44**, 14-26, (2016).
19 Chi, P. *et al.* Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. in *Proceedings of the 43rd International Symposium on Computer Architecture.* 27-39 (IEEE Press, 2016).
20 Qiao, X., Cao, X., Yang, H., Song, L. & Li, H. Atomlayer: a universal reRAM-based CNN accelerator with atomic layer computation. in *Proceedings of the 55th Annual Design Automation Conference.* 1-6 (ACM, 2018).
21 Ankit, A. *et al.* PUMA: A programmable ultra-efficient memristor-based accelerator for machine learning inference. in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems.* 715-731 (ACM, 2019).
22 Horowitz, M. Computing's energy problem (and what we can do about it). in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International.* 10-14 (IEEE, 2014).
23 Jouppi, N. P. *et al.* In-datacenter performance analysis of a tensor processing unit. in *Proceedings of the 44th Annual International Symposium on Computer Architecture.* 1-12 (ACM, 2017).
24 Jiang, H. *et al.* Sub-10 nm Ta Channel Responsible for Superior Performance of a HfO_2 Memristor. *Sci. Rep.* **6**, 28525, (2016).