A fully hardware-based memristive multilayer neural network

ENGINEERING

A fully hardware-based memristive multilayer neural network

Fatemeh Kiani, Jun Yin†, Zhongrui Wang‡, J. Joshua Yang†, Qiangfei Xia*

Memristive crossbar arrays promise substantial improvements in computing throughput and power efficiency through in-memory analog computing. Previous machine learning demonstrations with memristive arrays, however, relied on software or digital processors to implement some critical functionalities, leading to frequent analog/digital conversions and more complicated hardware that compromises the energy efficiency and computing parallelism. Here, we show that, by implementing the activation function of a neural network in analog hardware, analog signals can be transmitted to the next layer without unnecessary digital conversion, communication, and processing. We have designed and built compact rectified linear units, with which we constructed a two-layer perceptron using memristive crossbar arrays, and demonstrated a recognition accuracy of 93.63% for the Modified National Institute of Standard and Technology (MNIST) handwritten digits dataset. The fully hardware-based neural network reduces both the data shuttling and conversion, capable of delivering much higher computing throughput and power efficiency.

INTRODUCTION

Nonvolatile emerging memory devices such as memristors have been successfully adopted in hardware accelerators for machine learning (*1*–*15*). Organized into a crossbar array, these devices implement vector matrix multiplication (VMM), a computationally expensive operation within one step regardless of the array size, resulting in orders of magnitude higher computing throughput (*16*, *17*). The VMM is achieved by analog computing using physical laws, such as Ohm's law for multiplications and Kirchhoff's current law for summations (*5*–*7*, *17*), at the same site where the data are stored. This in-memory computing scheme hence substantially reduces power consumption by avoiding constant data shuttling as required in a traditional von Neumann architecture (*13*, *17*–*19*).

While the analog multilevel resistance of a memristive device has been successfully used as the synaptic weights in a neural network, most previous work still relied on, at least partially, software or digital processors to implement the hidden neurons (*5*–*7*, *14*). As a result, there is still frequent analog/digital (A/D) data conversion and back-and-forth data communication during computing, resulting in an unnecessary waste of resources. To unleash the potential of the emerging accelerators, a fully hardware-based neural network with efficient analog-based hardware neurons becomes imperative (*15*, *20*). Previous attempts to this end, however, have been limited to extremely small-scale arrays, large circuit area for hardware neurons, and low speed of the entire system (*21*–*23*).

In this work, we have designed and implemented a compact multichannel rectifying linear unit (ReLU) using off-the-shelf analog components. We further built a two-layer fully hardware-based perceptron with 64 ReLUs as the hidden neurons that connect two 128 × 64 memristive crossbar arrays, with which we have successfully achieved a 93.63% recognition accuracy on the classification of Modified National Institute of Standard and Technology (MNIST) (*24*) images. Compared with partial software hidden neurons, our analog-based fully hardware implementation is advantageous in power, area, and latency by removing the unnecessary data conversion, back-and-forth communication, and the corresponding circuit components.

RESULTS

Design and construction of the hardware

Our two-layer perceptron is composed of two memristive crossbar arrays representing the matrices of synaptic weights of each layer and the ReLUs as the activation functions in between (Fig. 1A). Analog signal generated from an in-house peripheral circuitry is inputted into the first crossbar array, the analog output (weighted sum along the columns) is sent to the hidden neurons (Fig. 1B), and the signal after is the input for the second crossbar array. To implement differential pairs that enable both positive and negative synaptic weights, both the original and the inverted signal out of the ReLUs are used as inputs (Fig. 1C). We used two individual chips, each contains 128 × 64 memristive crossbar arrays in a one-transistor one-resistance switch (1T1R) architecture (*25*–*27*), in this study to implement the VMM operations in the two layers (Fig. 1C). We fabricated the chips by the back-end-of-the-line integration of tantalum oxide–based memristors in our university laboratory with foundry-made transistor arrays (see Materials and Methods). The electrical performance of the individual cells and the arrays, as well as the conductance tuning schemes, can be found in fig. S1. We connected the two 1T1R arrays by a stack of printed circuit boards (PCBs) that hosts a total of 64 ReLU channels and 64 inverters built with off-the-shelf analog components (Fig. 1, D to F).

Each ReLU channel is composed of a half-wave current rectifier, a voltage follower, and an inverting amplifier, all built with operational amplifiers (Fig. 2A). The current rectifier generates a rectified output voltage directly from the input current, which avoids using an additional transimpedance amplifier for current-to-voltage conversion beforehand, resulting in a more compact design. The voltage follower is a unity gain buffer that is used to isolate the first stage

Department of Electrical and Computer Engineering, University of Massachusetts Amherst, Amherst, MA 01003, USA.
*Corresponding author. Email: qxia@umass.edu
†Present address: Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA 90007, USA.
‡Present address: Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong, China.

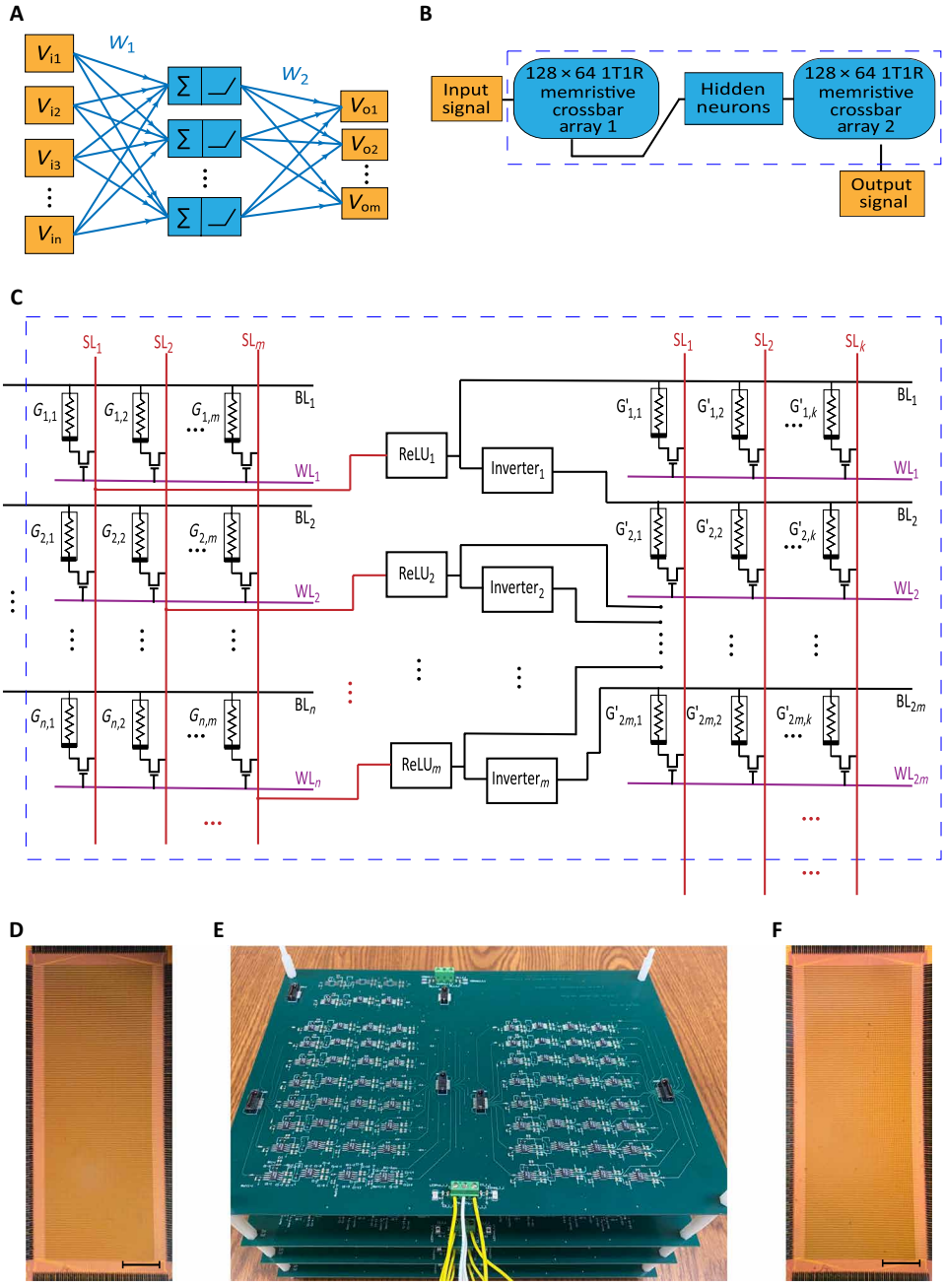

Fig. 1. The two-layer memristive perceptron built with hardware neurons and synapses. (**A**) Schematic of the perceptron, in which the hidden layer calculates the weighted sum of the inputs (v_i) and applies activation functions on the weighted sum. (**B**) Block diagram of the implemented perceptron. The weights [w_1 and w_2 as represented by arrows in (A)] are represented by the conductance of the memristive devices in the crossbar arrays. The activation functions are applied using the hidden neurons. (**C**) Circuit schematic of the perceptron. Crossbar arrays are composed of 1T1R cells, each connected in series between the selection line (SL) and bit line (BL) with the transistor gate connected to the word line (WL). Hidden neurons (ReLUs) and inverters provide both positive and negative inputs for the second layer. (**D**) An optical micrograph of the 1T1R array used as the left crossbar in (C) as the first layer. Scale bar, 1 mm. (**E**) A stack of four PCBs as the hardware neurons, each containing 16 channels of ReLU activation functions and 16 inverters to support 16 differential pairs for the second layer. (**F**) An optical micrograph of the 1T1R array used as the right crossbar in (C) as the second layer. Scale bar, 1 mm. Photo credit: Fatemeh Kiani, University of Massachusetts Amherst.

from the next one and provide stability for this stage. Because of its very high input resistance and very low output resistance, it prevents undesired impacts from the next stage that may interfere with its normal operation (*28*). The inverting amplifier provides a positive output voltage as needed by a ReLU activation function. It also scales down the output voltage to the range of 0 to 0.2 V to not change the conductance of the memristors in the second layer during inference (Fig. 2B). Figure 2C shows a representative output characteristic curve of the hardware ReLU. As expected, the output voltage is zero when the input current is negative, while linearly proportional to the

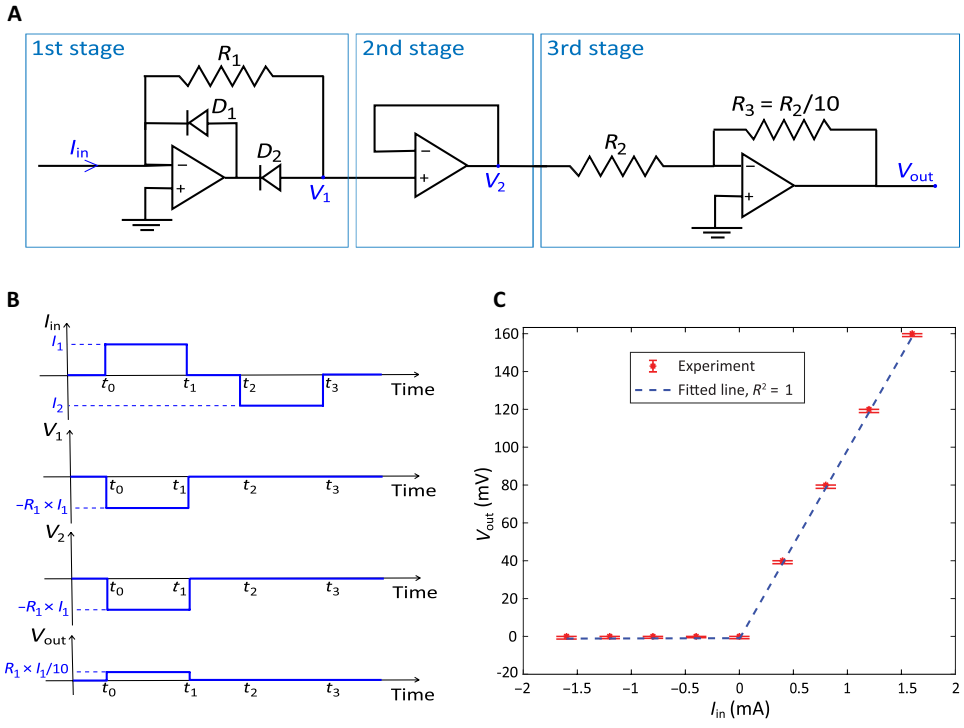

Fig. 2. Design and performance of the hardware ReLU. (**A**) The designed ReLU circuit. Each unit/channel consists of three stages: a current rectifier, a voltage follower, and an inverting amplifier, respectively (labeled in the blue boxes). The first stage rectifies the input current. The resistor on the feedback (R_1) is used to tune the gain of the ReLU, and the diodes (D_1 and D_2) are used to perform the rectification. The voltage follower, which is a unity gain voltage buffer, prevents the previous stage from loading by the next one. The third stage inverts its input to provide a positive voltage on the output as expected by a ReLU activation function. It also scales down the output by tuning the resistors R_2 and R_3. (**B**) Schematic of signal propagation from the input to the output. The input current (I_{in}) is rectified in the first stage and converted to the voltage using a gain equal to $-R_1$. The voltage output of the first stage (V_1) is duplicated to the output of the second stage (V_2) using the voltage follower and then inverted and scaled in the last stage (V_{out}) by multiplying with $-R_3/R_2$ (−0.1 in this design). (**C**) DC characteristic of the hardware ReLU. The output voltage is positive and linearly proportional to the input when the input current is positive and is around zero on the negative side. The output was measured 25 times for each input, and the error bar was plotted, which shows that the error from the expected output values is very small (less than 0.072% for the positive inputs and less than 1.5 mV for the negative ones). Linear regression was used to fit the measured data, and the R^2 was calculated to be 1, showing excellent linearity on the positive side.

current for positive inputs. We measured the output voltage 25 times for each input current, and the variation was less than 0.072% in each case on the positive side. We processed the experimental data and calculated the coefficient of determination (R^2) (*29*–*31*) to be 1.0, indicating excellent linearity on the positive side.

The relationship between the input current (I_{in}) and the output voltage (V_{out}) of the ReLU is defined as

$$V_{out} = \begin{cases} R_1 \times R_3/R_2 \times I_{in} & I_{in} \geq 0 \\ 0 & I_{in} < 0 \end{cases} \quad (1)$$

where R_1 is the feedback resistor of the current rectifier and R_3/R_2 is the scaling factor of the inverting amplifier, and the product $R_1 \times R_3/R_2$ is the gain of the ReLU.

The DC response shown in Fig. 2C is achieved when the gain of the ReLU was set to 100 with a scaling factor of 0.1. The gain is chosen upon the range of current flowing through the columns of the crossbar. By reducing the gain of the ReLU, the range of the current that can be sensed increases; however, the sensing resolution decreases (fig. S2). The delay time of the ReLU was measured to be 35 ns (see section S1 and fig. S3E), which is expected to decrease further by using more advanced manufacturing technology. For the current PCB implementation, the bottleneck lies in the data communication between different modules in the perceptron, and that between the perceptron and the off-chip peripheral circuits, which could be substantially reduced by integrating all the components on a single chip. Nevertheless, compared with the prior art (*22*, *23*), our hardware ReLU is stable for both positive and negative values of input current. Besides, because the current-to-voltage conversion and rectification are implemented in one single stage, the ReLUs use a smaller number of components, resulting in the potential reduction of circuit area and power consumption.

Training and inference methodology

To train and test the two-layer perceptron, we used electrical pulses with various amplitudes to represent the intensities of different pixels in an image. The pulses were inputs to the first layer along the rows of the first 1T1R array. The training was carried out in situ and partially on software using an in-house peripheral circuitry. During the training, we used software ReLU and SoftMax (*32*) activation functions in the hidden and output layer, respectively. To backpropagate the errors from the output to the input (*33*, *34*), we used cross-entropy

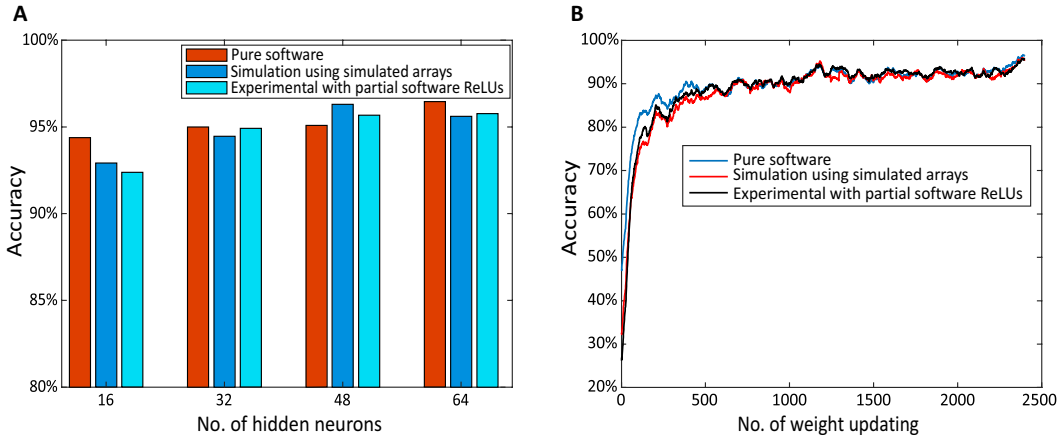

Fig. 3. In situ training of the two-layer networks. (**A**) Training accuracy increases with the number of hidden neurons. The network is trained using the MNIST dataset with a batch size of 50 and 2400 weight updates in total. Three different cases, namely, pure software training, simulation using simulated arrays (the hardware parameters were extracted and used to model the arrays), and experimental implementation with partial software ReLUs, are compared. (**B**) Training accuracy increases with the number of weight updating. The difference in the training accuracy for the three different cases narrows as well. During training, the weights in each array were updated, and the network accuracy was measured after each batch (50 images).

SoftMax as the loss function, which leads to faster convergence and lower error rates over the mean square error function in the classification applications (*32*, *35*, *36*). To calculate the required amount of weight updating, we used the root mean square propagation (RMSProp) optimizer because of its faster convergence over the stochastic gradient descent (SGD) (*24*, *32*, *37*–*43*). The weight updating of the memristor devices was realized using a one-shot blind update method (*44*), with one reset and one set pulse on the memristor electrode and the other synchronized pulse on the gate of the transistor.

The inference, on the other hand, was entirely conducted in the hardware. The input pulses were fed into the first memristive crossbar array, the computation results went through the hardware ReLUs and then to the second crossbar array, and the output current of the second layer was sensed and digitized for classification of the MNIST digits.

Classification of MNIST digits

To demonstrate the functionality, we have classified handwritten digits in the MNIST dataset, zero through nine, using the fully hardware-based two-layer perceptron. There are 128 input neurons, 64 hidden neurons, and 10 output neurons in the perceptron. We used the first 128×64 1T1R array as the first layer, while a 128 by 10 portion of the second 1T1R array was used as the second layer. During the computation, we used two neighboring cells on different rows in the 1T1R arrays to represent one synaptic weight. We cropped and downsampled the images in the MNIST datasets to 8 pixels by 8 pixels to accommodate the network dimension. We used all 60,000 training images in the MNIST dataset for the training and the 10,000 test images for the inference.

The training accuracy is dependent on several factors, including the device nonideality in the 1T1R crossbar arrays (see section S2 for details), the number of hidden neurons, and the number of weights updating. To understand the effects of these factors, we conducted both experimental and simulation training using a batch size of 50 during two training epochs and a total of 2400 weight updates. During the simulation, we used both a pure software model that assumes ideal devices and a practical model that considers the noise level and conductance dynamic range. While in the experimental training, we adopted partial software ReLUs. It was concluded that a larger number of hidden neurons has led to higher training accuracy (Fig. 3A). For example, by increasing the number of hidden neurons from 16 to 64, the training accuracy improved from 92.38 to 95.77% in the experimental case. The improvement could be resulted from the increased complexity and hence capability of the perceptron that better compensates the nonidealities of the devices in the network. Performing more weight updating also has a noticeable effect on the training accuracy, especially in the early stages (Fig. 3B). However, the increase of training accuracy slows down, and the difference between the experimental and simulation training results narrows down as the weight updating progresses. In other words, by performing more weight updating, the experimental accuracy reaches nearly the identical accuracy as that from the pure software simulation (less than 1% difference). It is also worth mentioning that the high training accuracy (95.77%) despite the device-to-device variations and existing imperfections of the arrays (see section S2 and fig. S1) is attributed to the in situ training scheme used in this study.

Similarly, the inference accuracy can be much improved by using a larger number of hidden neurons. To benchmark the performance, we used the 10,000 test images on our fully hardware perceptron, as well as the network with hidden neurons partially implemented in software. The inference accuracy is improved from 85.94 to 93.63% for our fully hardware perceptron with the number of hidden neurons increased from 16 to 64 (Fig. 4A). At the same time, the difference between the inference accuracy of the perceptron with hardware and partial software neurons is reduced from 1.27% to nearly zero. According to the confusion matrix (Fig. 4B), the most frequent misclassification occurred to the digit 2, mostly frequently misclassified as digit 0 or 3 (Fig. 4C). It is worth pointing out that the digits with higher classification errors are those that are difficult for human eyes to differentiate as well, possibly a side effect of the image downsampling that we did to accommodate the network dimension.

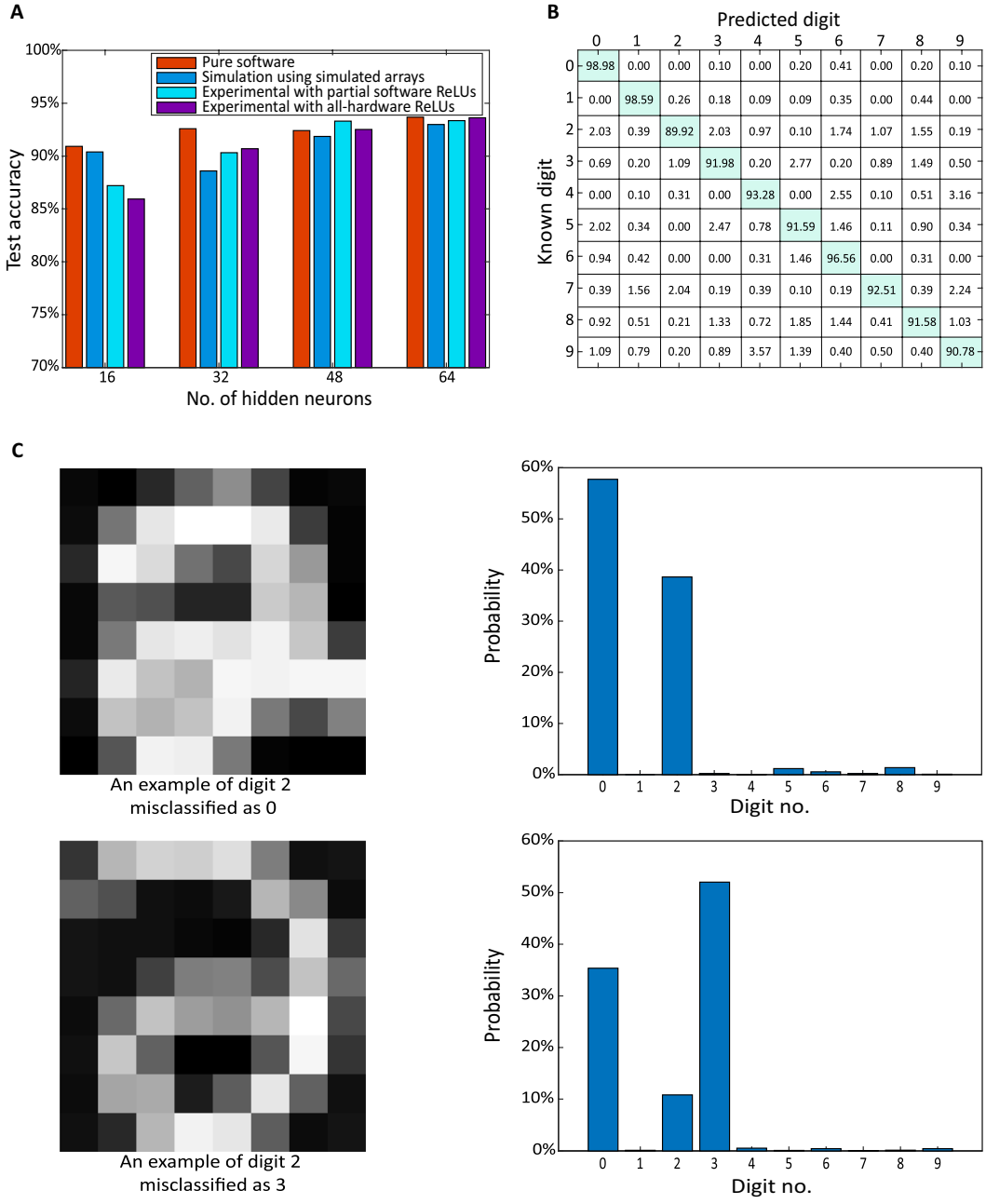

Fig. 4. Inference results of the two-layer hardware perceptron. (**A**) Inference accuracy increases with the number of hidden neurons. The accuracy is measured in different cases of using pure software, simulated arrays on simulation, experimental implementation using partial software hidden neurons, and experimental implementation using all-hardware hidden neurons. The improvement in test accuracy is more noticeable in the case of experimental implementation using all-hardware hidden neurons. (**B**) Confusion matrix of the inference accuracy using all-hardware hidden neurons. The most frequently misclassified images are digit 2. (**C**) Two different images of digit 2 (left column) have been applied as input, and on the basis of the output current, the corresponding probability of each digit (right column) has been calculated using a SoftMax function. The digit number with the highest probability shows the classification result. Both images have been misclassified, the first one as digit 0 and the second one as digit 3. These images are hard for human eyes to recognize as well.

Examination of nonidealities

To examine the effects of nonidealities of devices and built ReLUs on the functionality of the perceptron, we have modeled the nonidealities and tested their effects in classifying the MNIST digits in simulation. The linearity of conductance tuning during the gate voltage increments and decrements (see fig. S1B) is an important factor in the perceptron performance, meaning that large nonlinearity can substantially degrade the learning accuracy. To model the nonlinearity of devices in our perceptron, the relationship between the conductance of the devices and the gate voltage is considered to be exponential (see fig. S4A) (*45*). Then, these nonlinear models were used during the programming phase of training in our perceptron. The

results show that small nonlinearity affects the learning accuracy by less than 3%; however, large nonlinearities can substantially degrade the accuracy (fig. S4B).

Another nonideality that can affect the performance of the perceptron is the noise associated with our built ReLUs originating from the thermal noise of resistors as well as voltage and current noise of amplifiers (see fig. S5A). The total noise at the output of each channel of ReLU is 61.05 μV (see section S3 for the calculation). To examine the effect of noise, we conducted an experiment in simulation using our practical model perceptron by adding different values of noise to the output of ReLUs, and the learning accuracy during the inference was observed (fig. S5B), which shows that, by adding 61.05 μV of noise at the output, accuracy remains fixed. In addition, the results show that our perceptron is highly resilient to noise, and by adding a huge noise of 20 mV (10% of the output voltage), the accuracy drops by less than 5%.

There are two other factors that can affect the performance of the perceptron: mismatch and errors of ReLU gains. Gain of the ReLU is mostly determined by the feedback resistor of R_1 as shown in Fig. 2A. In the integrated design, it is feasible to fabricate resistors with 10 to 30% of error compared to the resistor nominal value (*46*) and less than 1.5% mismatch with an optimum layout configuration (*46*–*48*). We added up to 30% errors to the gain of ReLUs and examined the performance of the perceptron, which shows no accuracy degradation to the ideal case with zero error. We also performed Monte Carlo simulation (*49*), in which the mismatch of resistors is modeled by adding a normal distribution to the gain of ReLUs and the effect of the mismatch is observed in the system result. The SD of the mismatch was considered up to 3%, and the learning accuracy was measured 20 times for each SD. The mean value of accuracy was changed by 0.1%, and the maximum variation was 0.7%, which shows that our perceptron can tolerate 3% mismatch of resistors and 30% error of resistors without any substantial degradation in performance.

Estimation of power, area, and latency

To examine the efficiency in terms of power, area, and throughput, we compare the perceptron with analog hidden neurons with the same perceptron but with digital-based hidden neurons in encoding inputs using the amplitude of input pulses. The data flow for a perceptron with hidden neurons implemented using digital units involves A/D conversions and data communication between the hardware and those digital units for each layer (fig. S6 and table S1). The intensive use of power-hungry peripheral circuits such as analog-to-digital converters (ADCs)/digital-to-analog converters (DACs) and digital units adds up the power consumption and latency. However, in the case of using the analog-based hardware neurons, the analog signal is sent to the next layer of the perceptron through the ReLU boards directly. Hence, there is no need for peripheral analog circuits and digital-based data communication between the layers, substantially improving the power efficiency and the computing throughput. It is worth mentioning that in our measurements using our implemented perceptron, the latency of the hidden layer using analog-based hardware ReLUs is 35 ns, as mentioned previously, compared to the much higher latency that could be resulted from the hidden layer using digital-based ReLUs.

To fully realize the efficiency in terms of power and area, we have estimated the power consumption and area occupation of the designed analog-based hidden layer and the digital-based counterpart in the 65-nm complementary metal-oxide semiconductor (CMOS) technology node considering only the state-of-the-art DACs, ADCs, and amplifiers. The estimations show that using all hardware hidden neurons will improve power and area efficiencies by at least 32.8 and 5.64 times, respectively (see section S4 and table S2 for the estimation). It is worth mentioning that there is prior art (*50*) attempting to reduce the power consumption of the hidden layer by proposing a new ADC; however, the proposed ADC can only be used for a perceptron with the inputs encoded using pulse width and its resolution is in a trade-off with the energy consumption. The memristor devices were programmed either binary or 2 bit, and the hidden layer needs a digital controller, which induces extra latency.

DISCUSSION

The fully hardware-based two-layer perceptron inherits the benefits of using memristive devices as synapses for in-memory computing. In addition, using hardware neurons further improves the power and chip area efficiencies. To fully realize the efficiency in terms of power, area, and throughput, an integrated design of such an analog-based fully hardware perceptron is needed, on the basis of the proof-of-concept demonstration of this work. The integrated chip would have higher area efficiency and lower power consumption as compared to the board-level implementation. Furthermore, with all components built on a single chip, parasitic capacitance and resistance would also be much reduced (*51*), leading to substantially lower latency. Last but not least, the avoidance of data communication and A/D conversions between layers of perceptron would drastically reduce the latency and improve the computing throughput. This improvement in throughput can be substantial considering the computing overhead caused by digital computational units. Therefore, it is crucial to design and implement neurons, crossbar arrays, and the peripheral circuits, all on hardware using specifically designed analog components to enjoy the substantial benefits of memristive crossbars to the most in the perceptron implementation. Our fully analog-based perceptron is suited for in-sensor computing (*52*, *53*) that can further reduce the power consumption by processing the sensory data on-site.

To achieve a high recognition accuracy in the perceptron, it is necessary to fabricate memristive crossbar arrays with high yield and uniformity, meaning with a low number of defective devices on the array. It is also crucial to use a high number of hidden neurons because increasing the number of hidden neurons leads to higher accuracy. Moreover, achieving very high recognition accuracy for large input images need using larger arrays such that there would be no need to downsample the input images to accommodate the array size. In addition, it is worth mentioning that there is always a trade-off between the gain of the ReLUs and their bandwidth. Moreover, in the integrated design version where amplifiers are being designed with transistors, the mismatch of transistors and offset of amplifiers need to be studied. Therefore, it is necessary to design the ReLUs according to the desired specs.

In summary, we have designed and implemented a 64-channel hardware ReLU activation function using off-the-shelf analog components and built a two-layer fully hardware-based perceptron together with memristive crossbar arrays. The perceptron was used to classify digits from the MNIST dataset with a 93.63% recognition accuracy, comparable with that from a perceptron with the neurons partially implemented in software. The fully hardware perceptron

has advantages in power efficiency, area occupation, and computing throughput, which will be further enhanced with an implementation in integrated circuits.

MATERIALS AND METHODS

Hardware neuron and PCB design

We mounted off-the-shelf components including LTC6268 operational amplifiers (*54*) and 1N4148 surface mount type of diodes (*55*) to a four-layer PCB according to the ReLU design in Altium Designer. We used the top layer to route a part of the signals; the first internal layer was allocated to ground (GND) that serves as a fixed reference and current return paths for all the signals. We routed positive power traces on the second internal layer and the negative power traces and the other parts of the signals on the bottom layer. To keep the reference and current return paths fixed for all the signals, we poured multiple layers of polygon copper on the second internal layer above the signals of the bottom layer. We connected these copper layers to GND, resulting in high signal integrity and noise reduction (*56*, *57*).

Device fabrication and chip integration

The 1T1R cells and interconnection between cells were taped out in a commercial foundry with 2-μm technology, and we integrated the memristor devices on top of the CMOS substrates at University of Massachusetts Amherst (*6*). We cleaned the exposed metal vias on the CMOS substrates with argon plasma to remove the native oxide. Three different metal layers—~7-nm Ag, ~3-nm Ti, and ~50-nm Pd—were deposited sequentially by DC sputtering under an 8.8×10^{-7}–torr background vacuum, lifted off in acetone with a sonication process for 10 s at room temperature, and then annealed at 300°C for 40 min in N_2 atmosphere. The bottom electrodes (20-nm Pt with 2- to 2.5-nm Ti adhesion layer) were then patterned and evaporated onto the metal pads, followed by liftoff in acetone. A 5-nm Ta_2O_5 blank layer was sputtered in 90-W radio frequency power using 20 standard cubic centimeters per minute (sccm) Ar flow (background vacuum in the sputterer was better than 3.3×10^{-7} torr). The top electrodes (20-nm Ta and 20-nm Pt) were patterned by photolithography, deposited by DC sputtering, and lifted off in acetone at room temperature.

Inputs and weights conditioning

To map the intensities of image pixels to the inputs of the network, we conditioned the inputs. First, we downsampled the MNIST digits from 28 pixels by 28 pixels into 8 pixels by 8 pixels. To do so, we cropped the input images to 20 pixels by 20 pixels to keep only the central pixels. Then, we performed bicubic interpolation using MATLAB to rescale the images to 8 pixels by 8 pixels (*7*). Last, we mapped the intensities to the range of 0 to 0.2 V and applied them to the inputs using custom generated pulse signals, in which amplitude is equal to the scaled intensity. In addition, to enable implementing both positive and negative weights, we considered every two adjacent rows as a differential pair. To condition the weights difference after calculation by the RMSprop optimizer, we mapped the weights difference to the conductance difference of memristors by a factor of 10^{-4}; then, we programmed the memristors to the calculated conductance by tuning the gate of the corresponding 1T1R cell.

SUPPLEMENTARY MATERIALS

Supplementary material for this article is available at https://science.org/doi/10.1126/sciadv.abj4801

REFERENCES AND NOTES

1. M. Hu, C. E. Graves, C. Li, Y. Li, N. Ge, E. Montgomery, N. Davila, H. Jiang, R. S. Williams, J. J. Yang, Q. Xia, Memristor-based analog computation and neural network classification with a dot product engine. *Adv. Mater.* **30**, 1705914 (2018).
2. F. Cai, J. M. Correll, S. H. Lee, Y. Lim, V. Bothra, Z. Zhang, M. P. Flynn, W. D. Lu, A fully integrated reprogrammable memristor–CMOS system for efficient multiply–Accumulate operations. *Nat. Electron.* **2**, 290–299 (2019).
3. P. M. Sheridan, F. Cai, C. Du, W. Ma, Z. Zhang, W. D. Lu, Sparse coding with memristor networks. *Nat. Nanotechnol.* **12**, 784–789 (2017).
4. Y. Jeong, J. Lee, J. Moon, J. H. Shin, W. D. Lu, K-means data clustering with memristor networks. *Nano Lett.* **18**, 4447–4453 (2018).
5. P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, H. Qian, Fully hardware-implemented memristor convolutional neural network. *Nature* **577**, 641–646 (2020).
6. C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves, Z. Li, J. P. Strachan, P. Lin, Z. Wang, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, Q. Xia, Analogue signal and image processing with large memristor crossbars. *Nat. Electron.* **1**, 52–59 (2018).
7. C. Li, D. Belkin, Y. Li, P. Yan, M. Hu, N. Ge, H. Jiang, E. Montgomery, P. Lin, Z. Wang, W. Song, J. P. Strachan, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, Q. Xia, Efficient and self-adaptive in-situ learning in multilayer memristor neural networks. *Nat. Commun.* **9**, 2385 (2018).
8. J. Cui, Q. Qiu, Towards memristor based accelerator for sparse matrix vector multiplication, in *IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE, 2016), pp. 121–124.
9. M. A. Zidan, J. P. Strachan, W. D. Lu, The future of electronics based on memristive systems. *Nat. Electron.* **1**, 22–29 (2018).
10. T. F. Schranghamer, A. Oberoi, S. Das, Graphene memristive synapses for high precision neuromorphic computing. *Nat. Commun.* **11**, 5474 (2020).
11. C. X. Xue, Y. C. Chiu, T. W. Liu, T. Y. Huang, J. S. Liu, T. W. Chang, H. Y. Kao, J. H. Wang, S. Y. Wei, C. Y. Lee, S. P. Huang, J. M. Hung, S. H. Teng, W. C. Wei, Y. R. Chen, T. H. Hsu, Y. K. Chen, Y. C. Lo, T. H. Wen, C. C. Lo, R. S. Liu, C. C. Hsieh, K. T. Tang, M. S. Ho, C. Y. Su, C. C. Chou, Y. D. Chih, M. F. Chang, A CMOS-integrated compute-in-memory macro based on resistive random-access memory for AI edge devices. *Nat. Electron.* **4**, 81–90 (2020).
12. W. Zhang, B. Gao, J. Tang, P. Yao, S. Yu, M. F. Chang, H. J. Yoo, H. Qian, H. Wu, Neuro-inspired computing chips. *Nat. Electron.* **3**, 371–382 (2020).
13. J. Hung, X. Li, J. Wu, M. Chang, Challenges and trends in developing nonvolatile memory-enabled computing chips for intelligent edge devices. *IEEE Trans. Electron Devices* **67**, 1444–1453 (2020).
14. W. H. Chen, C. Dou, K. X. Li, W. Y. Lin, P. Y. Li, J. H. Huang, J. H. Wang, W. C. Wei, C. X. Xue, Y. C. Chiu, Y. C. King, C. J. Lin, R. S. Liu, C. C. Hsieh, K. T. Tang, J. J. Yang, M. S. Ho, M. F. Chang, CMOS-integrated memristive non-volatile computing-in-memory for AI edge processors. *Nat. Electron.* **2**, 420–428 (2019).
15. C. Li, D. Belkin, Y. Li, P. Yan, M. Hu, N. Ge, H. Jiang, E. Montgomery, P. Lin, Z. Wang, J. P. Strachan, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, Q. Xia, In-memory computing with memristor arrays, in *Proceedings of the IEEE International Memory Workshop (IMW)* (IEEE, 2018), pp. 1–4.
16. W. C. Wei, C. J. Jhang, Y. R. Chen, C. X. Xue, S. H. Sie, J. L. Lee, H. W. Kuo, C. C. Lu, M. F. Chang, K. T. Tang, A relaxed quantization training method for hardware limitations of resistive random access memory (ReRAM)-based computing-in-memory. *IEEE J. Explor. Solid-State Comput. Devices Circuits* **6**, 45–52 (2020).
17. B. Yan, B. Li, X. Qiao, C. X. Xue, M. F. Chang, Y. Chen, H. Li, Resistive memory-based in-memory computing: From device and large-scale integration system perspectives. *Adv. Intel. Syst.* **1**, 1900068 (2019).
18. Y. C. Chiu, Z. Zhang, J. J. Chen, X. Si, R. Liu, Y. N. Tu, J. W. Su, W. H. Huang, J. H. Wang, W. C. Wei, J. M. Hung, S. S. Sheu, S. H. Li, C. I. Wu, R. S. Liu, C. C. Hsieh, K. T. Tang, M. F. Chang, A 4-Kb 1-to-8-bit configurable 6T SRAM-based computation-in-memory unit-macro for CNN-based AI edge processors. *IEEE J. Solid State Circuits* **55**, 2790–2801 (2020).
19. C. X. Xue, W. H. Chen, J. S. Liu, J. F. Li, W. Y. Lin, W. E. Lin, J. H. Wang, W. C. Wei, T. Y. Huang, T. W. Chang, T. C. Chang, H. Y. Kao, Y. C. Chiu, C. Y. Lee, Y. C. King, C. J. Lin, R. S. Liu, C. C. Hsieh, K. T. Tang, M. F. Chang, Embedded 1-Mb ReRAM-based Computing-in-Memory macro with multibit input and weight for CNN-based AI edge processors. *IEEE J. Solid State Circuits* **55**, 203–215 (2020).
20. K. Berggren, Q. Xia, K. K. Likharev, D. B. Strukov, H. Jiang, T. Mikolajick, D. Querloiz, M. Salinga, J. R. Erickson, S. Pi, F. Xiong, P. Lin, C. Li, Y. Chen, S. Xiong, B. D. Hoskins, M. W. Daniels, A. Madhavan, J. A. Liddle, J. J. McClelland, Y. Yang, J. Rupp, S. S. Nonnenmann, K. T. Cheng, N. Gong, M. A. L. Montaño, A. A. Talin, A. Salleo, B. J. Shastri, T. F. de Lima, P. Prucnal, A. N. Tait, Y. Shen, H. Meng, C. R. Carmes, Z. Cheng, H. Bhaskaran, D. Jariwala, H. Wang, J. M. Shainline, K. Segall, J. J. Yang, K. Roy, S. Datta, A. Raychowdhury, Roadmap on emerging hardware and technology for machine learning. *Nanotechnology* **32**, 012002 (2020).

21. F. M. Bayat, M. Prezioso, B. Chakrabarti, H. Nili, I. Kataeva, D. Strukov, Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits. *Nat. Commun.* **9**, 2331 (2018).
22. O. Krestinskaya, K. N. Salama, A. P. James, Learning in memristive neural network architectures using analog backpropagation circuits. *IEEE Trans. Circuits Syst. I Regul. Pap.* **66**, 719–732 (2019).
23. B. Nursultan, O. Krestinskaya, Perceptron Linear Activation Function Design with CMOS-Memristive Circuits, in *Proceedings of the International Conference on Computing and Network Communications (CoCoNet)* (IEEE, 2018), pp. 231–234.
24. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998).
25. C. Yakopcic, T. M. Taha, R. Hasan, Hybrid crossbar architecture for a memristor based memory, in *Proceedings of the IEEE National Aerospace and Electronics Conference* (IEEE, 2014), pp. 237–242.
26. M. Zangeneh, A. Joshi, Performance and energy models for memristor-based 1T1R RRAM cell, in *Proceedings of the Great Lakes Symposium on VLSI*, (2012), pp. 9–14.
27. C. Nauenheim, *Integration of Resistive Switching Devices in Crossbar Structures* (Forschungszentrum Jülich GmbH, Zentralbibliothek, Verlag, 2010).
28. B. Razavi, *Fundamentals of Microelectronics* (Wiley, 2013).
29. R. G. D. Steel, J. H. Torrie, *Principles and Procedures of Statistics* (McGraw-Hill Book Company, 1960).
30. N. R. Draper, H. Smith, *Applied Regression Analysis* (Wiley, 1998).
31. A. P. Barten, The coefficient of determination for regression without a constant term, in *The Practice of Econometrics*, R. Heijmans, H. Neudecker, Eds. (Springer, Dordrecht, 1987), pp. 181–189.
32. I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, *Deep Learning* (The MIT Press, 2016).
33. D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).
34. Y. LeCun, D. Touresky, G. Hinton, T. Sejnowski, A theoretical framework for back-propagation, in *Proceedings of the 1988 Connectionist Models Summer School* (1988), vol. 1, pp. 21–28.
35. P. Golik, P. Doetsch, H. Ney, Cross-entropy vs. squared error training: A theoretical and experimental comparison. *Interspeech* **13**, 1756–1760 (2013).
36. L. Luo, Y. Xiong, Y. Liu, X. Sun, Adaptive gradient methods with dynamic bound of learning rate. arXiv:1902.09843 [cs.LG] (26 February 2019).
37. C. Igel, M. Hüsken, Improving the Rprop learning algorithm, in *Proceedings of the Second International ICSC Symposium on Neural Computation* (ICSC Academic Press, 2000), vol. 2000, pp. 115–121.
38. R. V. K. OReddy, B. S. Rao, K. P. Raju, Handwritten hindi digits recognition using convolutional neural network with rmsprop optimization, in *Second International Conference on Intelligent Computing and Control Systems (ICICCS)* (IEEE, 2018), pp. 45–51.
39. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization. arXiv:1412.6980 [cs.LG] (22 December 2014).
40. T. Tieleman, G. Hinton, Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* **4**, 26–31 (2012).
41. F. Zou, L. Shen, Z. Jie, W. Zhang, W. Liu, A sufficient condition for convergences of adam and rmsprop. *Proc. IEEE Conf. Comp. Vision Pattern Recog.* **1**, 11119–11127 (2019).
42. M. C. Mukkamala, M. Hein, Variants of RMSProp and Adagrad with logarithmic regret bounds. arXiv:1706.05507 [cs.LG] (17 June 2017).
43. S. Ruder, An overview of gradient descent optimization algorithms. arXiv:1609.04747 [cs.LG] (15 September 2016).
44. Z. Wang, C. Li, P. Lin, M. Rao, Y. Nie, W. Song, Q. Qiu, Y. Li, P. Yan, J. P. Strachan, N. Ge, N. McDonald, Q. Wu, M. Hu, H. Wu, R. S. Williams, Q. Xia, J. J. Yang, In situ training of feed-forward and recurrent convolutional memristor networks. *Nat. Mach. Intell.* **1**, 434–442 (2019).
45. W. Wu, H. Wu, B. Gao, P. Yao, X. Zhang, X. Peng, S. Yu, H. Qian, A methodology to improve linearity of analog RRAM for neuromorphic computing, in *IEEE Symposium on VLSI Technology* (IEEE, 2018), pp. 103–104.
46. Y. Cheng, The influence and modeling of process variation and device mismatch for analog/rf circuit design, in *Proceedings of the Fourth IEEE International Caracas Conference on Devices, Circuits and Systems* (IEEE, 2002), pp. D046.
47. W. Tian, P. Steinmann, E. Beach, I. Khan, P. Madhani, Mismatch characterization of a high precision resistor array test structure, in *Proceedings of the IEEE International Conference on Microelectronic Test Structures* (IEEE, 2008), pp. 11–16.
48. H. Thibieroz, P. Shaner, Z. C. Butler, Mismatch and flicker noise characterization of tantalum nitride thin film resistors for wireless applications, in *Proceedings of the 2001 International Conference on Microelectronic Test Structures* (2001), pp. 207–212.
49. H. Hung, V. Adzic, Monte carlo simulation of device variations and mismatch in analog intregrated circuits, in *Proceedings of the National Conference on Undergraduate Research (NCUR)* (2006), pp. 1–8.
50. Q. Liu, B. Gao, P. Yao, D. Wu, J. Chen, Y. Pang, W. Zhang, Y. Liao, C. X. Xue, W. H. Chen, J. Tang, Y. Wang, M. F. Chang, H. Qian, H. Wu, A fully integrated analog ReRAM based 78.4TOPS/W compute-in-memory chip with fully parallel MAC computing, in *Proceedings of the IEEE International Solid- State Circuits Conference* (IEEE, 2020), pp. 500–502.
51. J. Semmlow, *Circuits, Signals, and Systems for Bioengineers* (Academic Press, 2005).
52. F. Zhou, Y. Chai, Near-sensor and in-sensor computing. *Nat. Electron.* **3**, 664–671 (2020).
53. L. Mennel, J. Symonowicz, S. Wachter, D. K. Polyushkin, A. J. Molina-Mendoza, T. Mueller, Ultrafast machine vision with 2D material neural network image sensors. *Nature* **579**, 62–66 (2020).
54. Analog Devices, Inc., High speed op amps (bandwidth>50MHz), LTC6268/LTC6269 datasheet (Publication 62689f, 2014); www.analog.com/media/en/technical-documentation/data-sheets/62689f.pdf.
55. Vishay Intertechnology, Inc., Small signal fast switching diode, 1N4148W-G datasheet" (Publication 1N4148W-G,2020); www.vishay.com/docs/81139/1n4148w-g.pdf.
56. Texas Instruments, Inc., High-speed layout guidelines (Publication SCAA082A, 2017); www.ti.com/lit/an/scaa082a/scaa082a.pdf?ts=1610567082028&ref_url=https%253A%252F%252Fwww.google.com%252F.
57. J. Kim, H. Lee, J. Kim, Effects on signal integrity and radiated emission by split reference plane on high-speed multilayer printed circuit boards. *IEEE Trans. Adv. Packaging* **28**, 724–735 (2005).
58. W. Yan, R. Kolm, H. Zimmermann, A low-voltage low-power fully differential rail-to-rail input/output opamp in 65-nm CMOS, in *IEEE International Symposium on Circuits and Systems* (IEEE, 2008), pp. 2274–2277.
59. D. Li, Z. Zhu, R. Ding, Y. Yang, A 1.4-mW 10-bit 150-MS/s SAR ADC with nonbinary split capacitive DAC in 65-nm CMOS. *IEEE Transactions on Circuits and Systems II: Express Briefs* **65**, 1524–1528 (2018).
60. A. Bhide, O. E. Najari, B. Mesgarzadeh, A. Alvandpour, An 8-GS/s 200-MHz Bandwidth 68-mW ΔΣ DAC in 65-nm CMOS. *IEEE Trans. Circuits Syst II Express Briefs* **60**, 387–391 (2013).

Acknowledgments: The device fabrication and array integration work were performed, in part, at the Centre for Hierarchical Manufacturing (CHM), an NSF-sponsored Nanoscale Science and Engineering Center (NSEC), and, in part, at the Device Fabrication Facility in the Institute of Applied Life Sciences, both at the University of Massachusetts Amherst. **Funding:** This work is sponsored by Air Force Research Laboratory under agreement number FA8750-18-2-0122 and Air Force Office of Scientific Research under agreement number FA9550-19-1-0213. The U.S. government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory or the U.S. government. **Author contributions:** Q.X. conceived and supervised the project. F.K. designed the ReLUs, implemented the PCB boards, and tested the performance of the ReLU modules and the perceptron. J.Y. fabricated the 1T1R arrays. Z.W. helped with the electrical testing. Q.X. and F.K. wrote the manuscript. J.J.Y. and all other authors contributed to the analysis of the results and commented on the manuscript. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data needed to evaluate the conclusions in the paper are present in a DOI (doi:10.5061/dryad.w3r2280rf).

Submitted 14 May 2021
Accepted 4 October 2021
Published 24 November 2021
10.1126/sciadv.abj4801

ScienceAdvances

A fully hardware-based memristive multilayer neural network

Fatemeh KianiJun YinZhongrui WangJ. Joshua YangQiangfei Xia

Sci. Adv., 7 (48), eabj4801. • DOI: 10.1126/sciadv.abj4801

Science Advances (ISSN) is published by the American Association for the Advancement of Science. 1200 New York Avenue NW, Washington, DC 20005. The title *Science Advances* is a registered trademark of AAAS.