APPLIED SCIENCES AND ENGINEERING

One-step regression and classification with cross-point resistive memory arrays

Zhong Sun, Giacomo Pedretti, Alessandro Bricalli, Daniele Ielmini*

Machine learning has been getting attention in recent years as a tool to process big data generated by the ubiquitous sensors used in daily life. High-speed, low-energy computing machines are in demand to enable real-time artificial intelligence processing of such data. These requirements challenge the current metal-oxide-semiconductor technology, which is limited by Moore's law approaching its end and the communication bottleneck in conventional computing architecture. Novel computing concepts, architectures, and devices are thus strongly needed to accelerate data-intensive applications. Here, we show that a cross-point resistive memory circuit with feedback configuration can train traditional machine learning algorithms such as linear regression and logistic regression in just one step by computing the pseudoinverse matrix of the data within the memory. One-step learning is further supported by simulations of the prediction of housing price in Boston and the training of a two-layer neural network for MNIST digit recognition.

INTRODUCTION

Resistive memories, also known as memristors (*1*), including resistive switching memory (RRAM) and phase-change memory (PCM), are emerging as a novel technology for high-density storage (*2*, *3*), neuromorphic hardware (*4*, *5*), and stochastic security primitives, such as random number generators (*6*, *7*). Thanks to their ability to store analog values and to their excellent programming speed, resistive memories have also been demonstrated for executing in-memory computing (*8*–*17*), which eliminates the data transfer between the memory and the processing unit to improve the time and energy efficiency of computation. With a cross-point architecture, resistive memories can be naturally used to perform matrix-vector multiplication (MVM) by exploiting fundamental physical laws such as the Ohm's law and the Kirchhoff's law of electric circuits (*8*). Cross-point MVM has been shown to accelerate various data-intensive tasks, such as training and inference of deep neural networks (*11*–*14*), signal and image processing (*15*), and the iterative solution of a system of linear equations (*16*) or a differential equation (*17*). With a feedback circuit configuration, the cross-point array has been shown to solve systems of linear equations and calculate matrix eigenvectors in one step (*18*). Such a low computational complexity is attributed to the massive parallelism within the cross-point array and to the analog storage and computation with physical MVM. Here, we show that a cross-point resistive memory circuit with feedback configuration is able to accelerate fundamental learning functions, such as predicting the next point of a sequence by linear regression or attributing a new input to either one of two classes of objects by logistic regression. These operations are completed in just one step in the circuit, in contrast to the iterative algorithms running on conventional digital computers, which approach the solution with a polynomial time complexity.

RESULTS

Linear regression in one step

Linear regression is a fundamental machine learning (ML) model for regressive and predictive analysis in various disciplines, such as biology, social science, economics, and management (*19*–*21*). Logistic regression, instead, is a typical tool for classification tasks (*22*), e.g., acting as the last classification layer in a deep neural network (*23*, *24*). Because of their simplicity, interpretability, and well-known properties, linear and logistic regressions stand out as the most popular ML algorithms across many fields (*25*). A linear regression model is described by an overdetermined linear system given by

$$\boldsymbol{X}\boldsymbol{w} = \boldsymbol{y} \tag{1}$$

where $\boldsymbol{X}$ is an $N \times M$ matrix ($N > M$), $\boldsymbol{y}$ is a known vector with a size of $N \times 1$, and $\boldsymbol{w}$ is the unknown weight vector ($M \times 1$) to be solved. As the problem is overdetermined, there is typically no exact solution $\boldsymbol{w}$ to Eq. 1. The best solution of Eq. 1 can be obtained by the least squares error (LSE) approach, which minimizes the norm of error $\boldsymbol{\varepsilon} = \boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}$, namely, $\|\boldsymbol{\varepsilon}\| = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2$, where $\|\cdot\|_2$ is the Euclidean norm. The vector $\boldsymbol{w}$ minimizing $\|\boldsymbol{\varepsilon}\|$ is obtained by the pseudoinverse (*21*, *23*, *24*) [or Moore-Penrose inverse (*26*)] $\boldsymbol{X}^+$, given by

$$\boldsymbol{w} = \boldsymbol{X}^+\boldsymbol{y} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{2}$$

where $\boldsymbol{X}^T$ is the transpose of matrix $\boldsymbol{X}$.

To obtain the solution in Eq. 2, we propose a cross-point resistive memory circuit in Fig. 1A, where the matrix $\boldsymbol{X}$ is mapped by the conductance matrix $\boldsymbol{G}_X$ in a pair of cross-point arrays of analog resistive memories, the vector $\boldsymbol{y}$ corresponds to the opposite of the input current vector $\boldsymbol{i} = [I_1; I_2; \ldots; I_N]$, and $\boldsymbol{w}$ is represented by the output voltage vector $\boldsymbol{v} = [V_1; V_2; \ldots; V_M]$ ($M = 2$ in Fig. 1A).

For a practical demonstration of this concept, we adopted arrays of RRAM devices composed of a HfO_2 dielectric layer sandwiched between a Ti top electrode (TE) and a C bottom electrode (BE) (*27*). This type of RRAM device can be programmed to any arbitrary analog conductance within a certain range, thus allowing to represent the matrix elements X_{ij} of the matrix $\boldsymbol{X}$ with sufficient accuracy (*18*). Representative analog conductance levels were programmed by controlling the compliance current during the set transition, as shown in fig. S1. The cross-point arrays are connected within a nested feedback loop (*28*) by N operational amplifiers (OAs) from the left array to the right array and M OAs from the right array to the left array. Briefly, the first set of OAs gives a negative transfer

Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano and IU.NET, Piazza L. da Vinci 32, 20133 Milano, Italy.
*Corresponding author. Email: daniele.ielmini@polimi.it

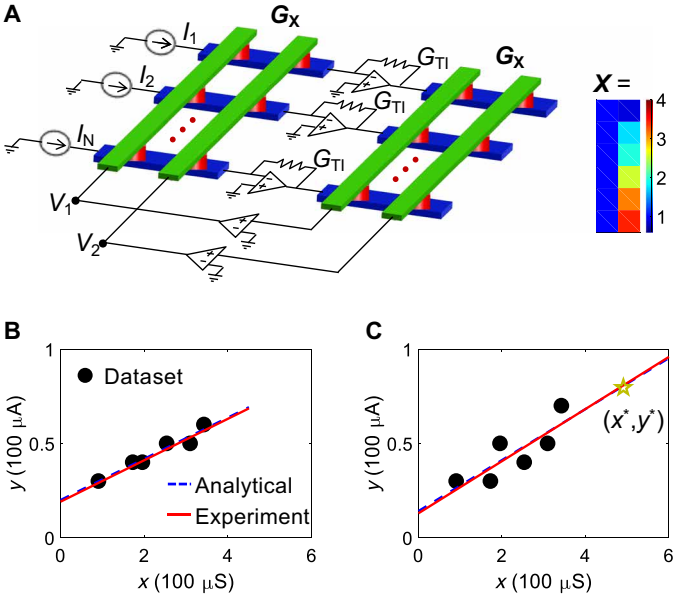

Fig. 1. Linear regression circuit and experiments. (**A**) Schematic illustration of the cross-point circuit for solving linear regression with the pseudoinverse method. The conductance transformation unit is $G_0 = 100$ μS. The feedback conductance G_{TI} of TIA is equal to G_0. A representative matrix $\boldsymbol{X}$ for simple linear regression of six data points is also shown. (**B**) Linear regression of a six-point dataset defined by the second column of $\boldsymbol{X}$ in (A) on the x axis and the input currents on the y axis. The figure also shows the analytical and experimental regression lines, and the latter being obtained as the measured voltages $\boldsymbol{v}$ in the cross-point circuit as regression weights. (**C**) A second six-point regression experiment with the same vector $\boldsymbol{X}$ in (A) and a different set of input currents. A new input value $x^* = 4.91$ was stored in an additional line of the left cross-point array, thus enabling the one-step prediction along a sequence. The measured prediction $y^* = 0.727$ is consistent with experimental and analytical regression lines.

of the current, while the second one gives a positive transfer, resulting in an overall negative feedback, hence stable operation of the circuit with virtual ground inputs of all OAs. A detailed analysis of the circuit stability is reported in text S1.

According to Ohm's law and Kirchhoff's law in Fig. 1A, the input currents at the OAs from the left cross-point array are $\boldsymbol{G_X v} + \boldsymbol{i}$; thus, the output voltages applied to the right cross-point array are $\boldsymbol{v_r} = -\frac{(\boldsymbol{G_X v} + \boldsymbol{i})}{G_{TI}}$, where G_{TI} is the feedback conductance of transimpedance amplifiers (TIAs). The right cross-point array operates another MVM between the voltage vector $\boldsymbol{v_r}$ and the transpose conductance matrix $\boldsymbol{G_X}^T$, resulting in a current vector $-\boldsymbol{G_X}^T\frac{(\boldsymbol{G_X v} + \boldsymbol{i})}{G_{TI}}$, which is forced to zero at the input nodes of the second set of OAs, namely

$$\boldsymbol{G_X}^T(\boldsymbol{G_X v} + \boldsymbol{i}) = \boldsymbol{0} \tag{3}$$

The steady-state voltages $\boldsymbol{v}$ at the left array are thus given by

$$\boldsymbol{v} = -\left(\boldsymbol{G_X}^T\boldsymbol{G_X}\right)^{-1}\boldsymbol{G_X}^T\boldsymbol{i} \tag{4}$$

which is Eq. 2 with $\boldsymbol{G_X}$, $\boldsymbol{i}$, and $\boldsymbol{v}$ representing $\boldsymbol{X}$, $-\boldsymbol{y}$, and $\boldsymbol{w}$, respectively. The cross-point array circuit of Fig. 1A thus solves the linear regression problem in just one step.

The circuit of Fig. 1A was implemented in hardware using RRAM devices arranged within a cross-point architecture on a printed circuit board (see Materials and Methods and fig. S2). As a basic model, we considered the simple linear regression of points (x_i, y_i), where $i = 1,2, \ldots, N$, to be fitted by a linear model $w_0 + w_1 x_i = y_i$, where w_0 and w_1 are the intercept with axis y and the slope, respectively, of the best fitting line. To solve this problem in hardware, we encoded the matrix $\boldsymbol{X}$

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix} \tag{5}$$

in the cross-point arrays. A column of discrete resistors with $G = 100$ μS was used to represent the first column of $\boldsymbol{X}$ in Eq. 5, which is identically equal to 1. The second columns of both arrays were implemented with reconfigurable RRAM devices. A total number of $N = 6$ data points were considered, with each x_i implemented as a RRAM conductance with unit 100 μS. The unit of conductance was chosen according to the range of linear conduction of the device (*18*), thus ensuring a good computation accuracy. Other aspects such as the current limit of the OAs and the power consumption should also be considered to select the best memory devices in the circuit. Although the conductance values in the two cross-point arrays should be identical, some mismatch can be tolerated for practical implementations (text S2). A program/verify technique was used to minimize the relative error (less than 5%) between the values of $\boldsymbol{X}$ in the two cross-point arrays (fig. S3). The data ordinates $-\boldsymbol{y}$ were instead applied as input currents. The input currents should be kept relatively small so that the resulting output signal is low enough to prevent disturbance of the device states in the stored matrix.

Given the matrix $\boldsymbol{X}$ stored in the cross-point arrays and an input current vector $\boldsymbol{y}$, the corresponding linear system was then solved by the circuit in one step. Figure 1B shows the resulting dataset for an input current vector $\boldsymbol{i} = [0.3; 0.4; 0.4; 0.5; 0.5; 0.6]I_0$ with $I_0 = 100$ μA to align with the conductance transformation unit. Figure 1B also shows the regression line, obtained by the circuit output voltages representing weights w_0 and w_1. The comparison with the analytical regression line shows a relative error of −4.86 and 0.82% for w_0 and w_1, respectively. The simulated transient behavior of the circuit is shown in fig. S4, evidencing that the linear regression weights are computed within about 1 μs. By changing the input vector, a different linear system was formed and solved by the circuit, as shown in Fig. 1C for $\boldsymbol{i} = [0.3; 0.3; 0.5; 0.4; 0.5; 0.7]I_0$. The result evidences that a more scattered dataset can also be correctly fitted by the circuit.

The cross-point circuit also naturally yields the prediction of the value y^* in response of a new point at position x^*. This is obtained by adding an extra row in the left cross-point, where an additional RRAM element is used to implement the new coordinate x^* (fig. S5). The results are shown in Fig. 1C, indicating a prediction by the circuit, which is only 1% smaller compared to the analytical prediction. Figure S6 reports more linear regression and prediction results of various datasets. Linear regression with two independent variables was also demonstrated by a cross-point array of three columns, with results shown in fig. S7. These results support the cross-point circuit for the solution of linear regression models in various dimensions. The linear regression concept can also be extended to nonlinear regression models, e.g., polynomial regression (*29*), to better fit a dataset and thus make better predictions. By loading the polynomial

terms in cross-point arrays, the circuit can also realize polynomial regression in one step (text S3 with fig. S8).

Logistic regression

Logistic regression is a binary model that is extensively used for object classification and pattern recognition. Different from linear regression, which is a fully linear model, logistic regression also includes a nonlinear sigmoid function to generate the binary output. A logistic regression model can be viewed as the single-layer feed-forward neural network in Fig. 2A. Here, the weighted summation vector $\boldsymbol{s}$ of input signals to the nonlinear neuron is given by

$$\boldsymbol{s} = \boldsymbol{X}\boldsymbol{w} \tag{6}$$

where $\boldsymbol{X}$ is the matrix containing the independent variable values of all samples and $\boldsymbol{w}$ is the vector of the synaptic weights. The neuron outputs are thus simply given by the vector $\boldsymbol{y} = f(\boldsymbol{s})$, where f is the nonlinear function of the neuron. To compute the weights of a logistic regression model with a sample matrix $\boldsymbol{X}$ and a label vector $\boldsymbol{y}$, the logit transformation can be first executed (*30*). By applying the inverse of sigmoid function, the label vector $\boldsymbol{y}$ is converted to a summation vector $\boldsymbol{s}$, namely, $\boldsymbol{s} = f^{-1}(\boldsymbol{y})$. As a result, the logistic regression is reduced to a linear regression problem, where the weights can be obtained in one step by the pseudoinverse concept

$$\boldsymbol{w} = \boldsymbol{X}^{+}\boldsymbol{s} \tag{7}$$

For simplicity, we assumed that the nonlinear neuron function is instead a step function and that the summation vector $\boldsymbol{s}$ in Fig. 2A is binarized according to

$$s_i = \begin{cases} a, if\, y_i = 1 \\ -a, if\, y_i = 0 \end{cases} \tag{8}$$

where a is a positive constant for regulating the output voltage in the circuit. After this transformation, the weights can be computed directly with the pseudoinverse circuit of Fig. 1A.

Figure 2B shows a set of six data points with coordinates (x_1, x_2) divided into two classes, namely, $y = 0$ (open) and $y = 1$ (full). Figure 2C shows the matrix $\boldsymbol{X}$ where the first column is equal to **1**, while the other columns represent the coordinates x_1 and x_2 of the dataset.

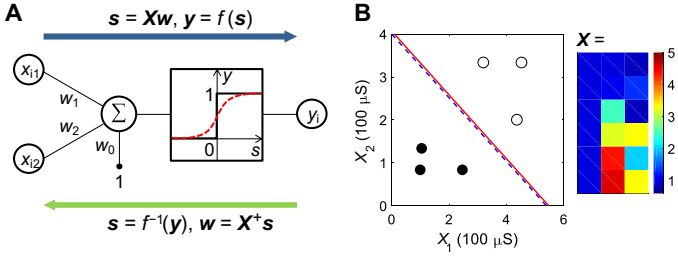

Fig. 2. Logistic regression experiments. (**A**) Illustration of a logistic regression model, consisting of the summation of weighted input signals being processed by a nonlinear activation function such as the sigmoid function (dash line) or the step function (solid line). The backward logit transformation is indicated by the bottom arrow. (**B**) A logistic regression of six data points divided in two classes. The input matrix $\boldsymbol{X}$ is also shown, including a first column of discrete resistors and a second and third columns storing the independent variables x_1 and x_2. The regression lines obtained from analytical and experimental results are also shown. The line provides the boundary line for data classification.

The sample matrix $\boldsymbol{X}$ was mapped in the two cross-point arrays of Fig. 1A, and input current was applied to each row to represent $\boldsymbol{s}$ with $a = 0.2$, according to Eq. 8. The circuit schematic is reported in fig. S9 together with experimental results and relative errors of logistic regression. The simulated transient behavior of the circuit is shown in fig. S10, with a computing time around 0.6 μs. The output voltage yields the weights $\boldsymbol{w} = [w_0; w_1; w_2]$ with $s = w_0 + w_1x_1 + w_2x_2 = 0$ representing the decision boundary for classification, where $s \geq 0$ indicates the domain of class "1" and $s < 0$ the domain of class "0". This is shown as a line in Fig. 2B, displaying a tight agreement with the analytical solution. The cross-point circuit enables a one-step solution of logistic regression with datasets of various dimensionalities and sizes accommodated by the cross-point arrays. Similar to linear regression, the circuit can also provide one-step classification of any new (unlabeled) point, which is stored in a grounded additional row of the left cross-point array. The current flowing in the row yields the class of the new data point. Although here we consider two cases containing only positive independent variable values for linear/logistic regression, datasets containing negative values can also be addressed by simply translating the entire data to be positive, as explained in fig. S11.

Linear regression of Boston housing dataset

While the circuit capability has been demonstrated in experiment for small models, the matrix size is an obvious concern that needs to be addressed for real-world applications. To study the circuit scalability, we considered a large dataset, namely, the Boston housing price list for linear regression (*31*, *32*). The dataset collects 13 attributes and the prices of 506 houses, 333 of which are used for training the linear regression model, while the rest are used for testing the model. The attributes are summarized in the text S4. We performed linear regression with the training set to compute the weights with the cross-point circuit and applied the regression model to predict house prices of the test set.

Figure 3A shows the matrix $\boldsymbol{X}$ for the training set, including a first column of 1, the other columns recording the 13 attributes, and the input vector $\boldsymbol{y}$, representing the corresponding prices. The matrix $\boldsymbol{X}$ was rescaled to make the conductance values in cross-point arrays uniform, and the vector $\boldsymbol{y}$ was also scaled down to prevent excessive output voltage $\boldsymbol{w}$ (see fig. S12). We simulated the linear regression circuit with SPICE (Simulation Program with Integrated Circuit Emphasis; see Materials and Methods), where the RRAM devices were assumed to accurately map the matrix values within 8-bit precision. Figure 3B shows the calculated $\boldsymbol{w}$ obtained from the output voltage in the simulated circuit, with the relative errors remaining within ±1%, thus demonstrating the good accuracy and scalability of the circuit.

Figure 3C shows the obtained regression results compared with the real house prices of the training set. A standard deviation (SD) σ_P of \$4733 is obtained from SPICE simulations, which is in line with the analytical solution σ_P' = \$4732. Figure 3D shows the predicted prices of the test set compared with the real prices. The SD from the circuit simulation is σ_P = \$4779, in good agreement with the analytical results σ_P' = \$4769. The resulting SD is only slightly larger than the training set, which supports the ability for generalization of the model. One-step price prediction for test samples is possible by storing the unlabeled attributes in additional rows of the left cross-point array and measuring the corresponding currents, as indicated in fig. S13.

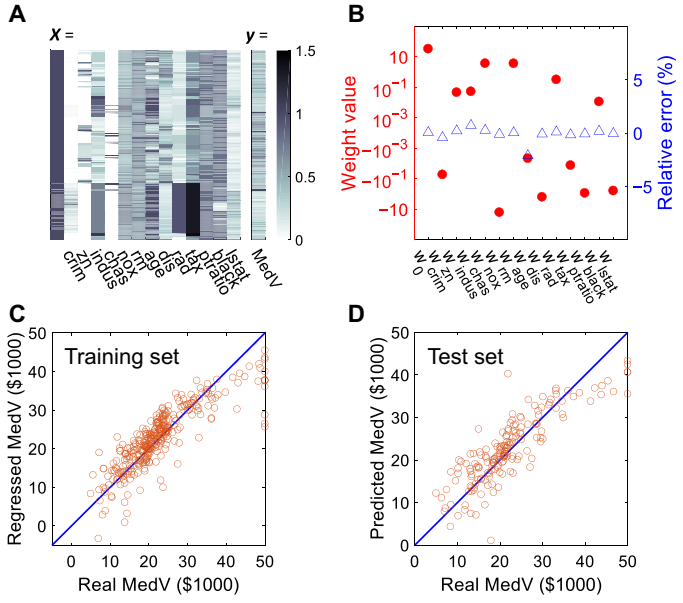

Fig. 3. Linear regression of the Boston housing dataset. (**A**) Matrix $\boldsymbol{X}$ including the 13 attributes for the 333 houses in the training set and input vector $\boldsymbol{y}$ of house prices. The same color bar was used for clarity, while the conductance and current units are assumed equal to 10 μS and 10 μA, respectively. (**B**) Calculated weights of the linear regression obtained by simulation of the cross-point circuit and the relative errors with respect to the analytical results. (**C**) Correlation plot of the regression price of the training samples obtained by the simulated weights, as a function of the real dataset price. The small SD σ_P = $4733 supports the accuracy of the regression. (**D**) Same as (C), but for the test samples. A slightly larger SD σ_P = $4779 is obtained.

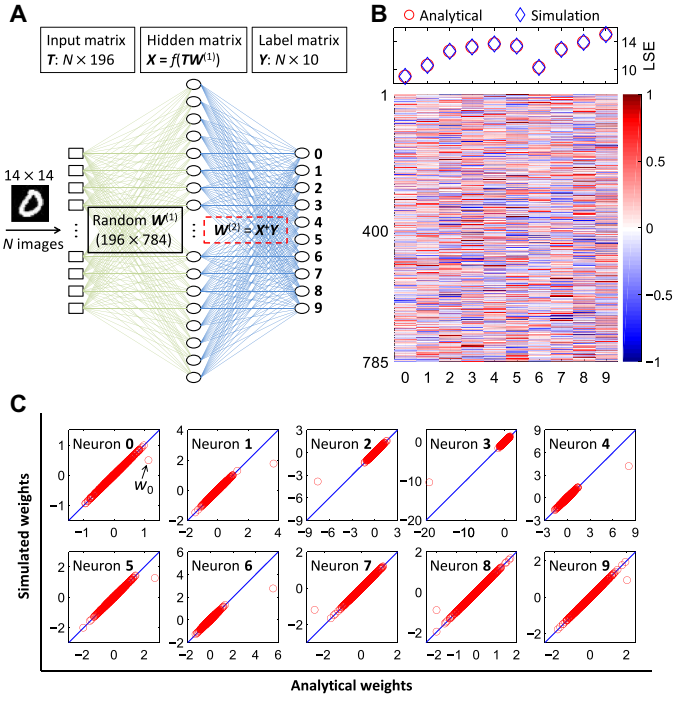

Fig. 4. Training of a two-layer neural network for MNIST digit recognition. (**A**) Illustration of the two-layer neural network, where the first-layer weights are random, while the second-layer weights are computed by the pseudoinverse method in circuit simulations. (**B**) Color plot of the second-layer weight matrix $\boldsymbol{W}^{(2)}$ obtained by circuit simulations. Each column contains the weights of synapses connected to an output neuron and was computed in one step by the cross-point circuit. As a result, only 10 operations were needed to train the network. The LSEs of simulated and analytical weights are also shown for each neuron. (**C**) Correlation plots of the simulated weights as a function of the analytical weights for each of the 10 output neurons. Only the bias weight w_0 shows a deviation from the analytical results, although not affecting the recognition accuracy.

Two-layer neural network training

Logistic regression is widely used in the last fully connected classification layer in deep neural networks. The cross-point circuit thus provides a hardware acceleration of the computation of the last-layer weights for training of a neural network. To test the cross-point circuit as an accelerator for training neural networks, we considered the two-layer perceptron in Fig. 4A, where the first-layer weights are set randomly (*33*, *34*), while the second-layer weights can be obtained by the pseudoinverse method in the cross-point circuit. Note that a standard technique to train a two-layer neural network is the backpropagation algorithm which reduces the squares error iteratively (*35*). In contrast to iterative backpropagation, the pseudoinverse approach can reach the LSE solution in just one step, thus providing a fast and energy-efficient acceleration of network training.

As a case study for neural network training, we adopted the Modified National Institute of Standards and Technology (MNIST) dataset (*36*). To reduce the circuit size in the simulations, we used only 3000 of 50,000 samples to train the neural network. Also, to provide an efficient fan-out (for instance, four) for the first layer (*34*), the image size was down-sampled to be 14 × 14, resulting in a network of 196 input neurons, 784 hidden neurons, and 10 output neurons for the classifications of the digits from 0 to 9. The training matrix $\boldsymbol{T}$ is with a size of 3000 × 196 and the first-layer weights $\boldsymbol{W}^{(1)}$ were randomly generated in the range between −0.5 and 0.5 with a uniform distribution (fig. S14). The matrix $\boldsymbol{X}$ can thus be obtained by

$$\boldsymbol{X} = f\left(\boldsymbol{T}\boldsymbol{W}^{(1)}\right) \tag{9}$$

while the weights of the second layer $\boldsymbol{W}^{(2)}$ can be obtained by the pseudoinverse model of Eq. 2, with $\boldsymbol{Y}$ containing all the known labels of training samples transformed according to Eq. 8 with a = 0.05. For each training sample, the neuron corresponding to the digit is labeled 1, while the other nine neurons are 0. Note that the matrix $\boldsymbol{X}$ results from the output of a sigmoid function of hidden neuron and is restricted in the range between 0 and 1 (fig. S15).

Figure 4B shows the second-layer weights $\boldsymbol{W}^{(2)}$ obtained by the simulation of the cross-point circuit, where $\boldsymbol{X}$ was stored in the RRAM devices, and each column of matrix $\boldsymbol{Y}$ was applied as input current. The weights were obtained in 10 steps, one for each classification output (from digit 0 to digit 9). With the computed weights $\boldsymbol{W}^{(2)}$, the network can recognize 500 handwritten digits with accuracy of 94.2%, which is identical to the analytical pseudoinverse solution. For the whole test set (10,000 digits), the recognition accuracy is 92.15% using the simulated $\boldsymbol{W}^{(2)}$, compared to 92.14% using the analytical solution. The cross-point array can thus be used to accelerate the training of typical neural networks with ideal accuracy. The computed weights can then be stored in one or more open-loop cross-point array for accelerating the neural network in the inference mode by exploiting in-memory MVM (fig. S16) (*8*, *11*, *12*).

Figure 4B also shows the LSEs obtained from both the circuit simulation and analytical study. Note that the LSEs are different among the 10 digits due to the dependence of LSE on weight values

(text S5). Figure 4C shows the simulated weights as a function of the analytical values for each output neuron, showing a good consistency except for the bias weight w_0. The bias acts as a regulator to the summation of an output neuron; thus, the deviated bias weight guarantees that the simulated LSE is close to the analytical one in Fig. 4B. It should be noted that, although a random $\boldsymbol{W}^{(1)}$ was assumed in this study, $\boldsymbol{W}^{(1)}$ can be further optimized by gradient descent methods (*37*) to improve the accuracy. The same approach might be applied to pretrained deep networks by the concept of transfer learning (*38*), thus enabling the one-step training capability for a generalized range of learning tasks.

DISCUSSION

Although the cross-point circuit is inherently accurate and scalable, the imperfections of RRAM devices such as conductance discretization and stochastic variation (*18*) might affect the solution. To study the impact of these issues on the solution accuracy, we assumed a RRAM model with 32 discrete conductance levels, including 31 uniformly spaced levels and one deep high-resistance state, which is achievable in many resistive memory devices (*39*–*41*). The ratio between the maximum conductance G_{max} and the minimum conductance G_{min} is assumed to be $\frac{G_{max}}{G_{min}} = 10^3$, in line with previous reports (*42*, *43*). To describe conductance variations, we assumed an SD $\sigma = \Delta G/6$, $\Delta G/4$, or $\Delta G/2$, where ΔG is the nominal difference between two adjacent conductance levels. The simulation results for the Boston housing benchmark (fig. S17) shows that the resulting regression and prediction remain accurate for all cases. For the worst case ($\sigma = \Delta G/2$), the SD σ_P of training set is equal to \$4756 compared with the ideal result of \$4732. The σ_P of test set for prediction is even closer to the ideal one, namely, \$4765 compared with \$4769. These results highlight the suitability of the cross-point resistive memory circuit for ML tasks, where the device variations can be tolerated for regression, prediction, and classification.

Another concern for large-scale circuits is the parasitic wire resistance. To study its impact on the accuracy of linear regression for Boston housing dataset, we adopted interconnect parameters at a 65-nm technology obtained from the International Technology Roadmap for Semiconductors table (*44*), together with the RRAM model. The results in fig. S18 show an increased σ_P for both regression and prediction, with the latter being less notable, which is consistent with the impact of device variation. Specifically, the σ_P of prediction becomes merely \$4809 compared with the ideal \$4769, thus supporting the robustness of the linear regression circuit for predictive analysis.

The circuit stability analysis in text S1 reveals that the poles of the system all lie in the left half plane; thus, the circuit is stable, and the computing time is limited by the bandwidth corresponding to the first pole, which is the minimal eigenvalue (or real part of eigenvalue) λ_{min} (absolute value) of a quadratic eigenvalue problem (*45*). As λ_{min} becomes larger, the computation of the circuit gets faster, with no direct dependence on the size of the dataset. To support this scaling property of the circuit speed, we have simulated the transient dynamics of linear regression of the Boston housing dataset and its subsets for increasing size of the training samples (fig. S19). The results show that the computing time may even decrease as the number of samples increases, which can be explained by the different λ_{min} of the datasets (fig. S20). These results evidence that the time complexity of the cross-point circuit for linear regression substantially differs from its counterparts of classical digital algorithms, with a potential of approaching size-independent time complexity to hugely speed up large-scale ML problems. Note that as the circuit size increases, a larger current is also required to sustain the circuit operation, which might be limited by the capability of the OAs. To control the maximum current consumption in the circuit, the memory element should be carefully optimized by materials and device engineering (*46*) or by advanced device concepts such as electrochemical transistor (*47*, *48*) to provide a low-conductance implementation. The impact of device variations and the energy efficiency of the circuit are studied for the two-layer neural network for MNIST dataset training (text S6 with fig. S21). The results support the robustness of the circuit against device variations for classification applications, and an energy efficiency of 45.3-tera operations per s per Watt (TOPS/W), which is 19.7 and 6.5 times better than the Google's tensor processing unit (*49*) and a highly optimized application-specific integrated circuit system (*50*), respectively.

In conclusion, the cross-point circuit has been shown to provide a one-step solution to linear regression and logistic regression, which is demonstrated in experiments with RRAM devices. The one-step learning capability relies on the high parallelism of analog computing by physical Ohm's law and Kirchhoff's law within the circuit and physical iteration within the nested feedback architecture. The scalability of the cross-point computing is demonstrated with large problems, such as the Boston housing dataset and the MNIST dataset. The results evidence that in-memory computing is remarkably promising for accelerating ML tasks with high latency/energy performance in a wide range of data-intensive applications.

MATERIALS AND METHODS

RRAM device fabrication

The RRAM devices in this work used a 5-nm HfO_2 thin film as the dielectric layer, which was deposited by e-beam evaporation on a confined graphitic C BE. Without breaking the vacuum, a Ti layer was deposited on top of the HfO_2 layer as TE. The forming process was operated by applying a dc voltage sweep from 0 to 5 V, where the voltage was applied to the TE and the BE was grounded. After the forming process, the set and reset transitions took place under positive and negative voltages applied to the TE, respectively.

Circuit experiment

For all the experiments, the devices were arranged in the cross-point configuration on a custom-printed circuit board (PCB; see fig. S2), and an Agilent B2902A Precision Source/Measure Unit was used to program the devices to different conductance states. Linear and logistic regression experiments were carried out on a custom PCB with OAs of model AD823 (Analog Devices) for the negative-feedback amplifiers (NFA) and OP2177 (Analog Devices) for positive-feedback amplifiers (PFA). RRAM devices of left matrix were connected with the BE to the NFAs' inverting input nodes and with the TE to the PFAs' output terminals. RRAM devices of right matrix were connected with the BE to the PFAs' non-inverting input nodes and with the TE to the NFAs' output terminals. A BAS40-04 diode is connected between every amplifier and ground to limit the voltages within ±0.7 V, avoiding conductance changes of RRAM devices.

All the input signals were given by a four-channel arbitrary waveform generator (Aim-TTi TGA12104) and applied to fixed input resistors, which were connected between the input and the NFAs' inverting-input nodes. The PFAs' output voltages were monitored by an oscilloscope (LeCroy Wavesurfer 3024). The board was powered by a BK Precision 1761 dc power supply.

SPICE simulation

Simulations of the cross-point circuit for Boston housing case and MNIST training were carried out using LTSPICE (www.linear.com/solutions/1066). Linear resistors with defined conductance values were used to map a matrix in the cross-point arrays. A universal op-amp model was used for all OAs, while PFA and NFA have different parameters.

SUPPLEMENTARY MATERIALS

Supplementary material for this article is available at http://advances.sciencemag.org/cgi/content/full/6/5/eaay2378/DC1

Text S1. Analysis of circuit stability
Text S2. Analysis of twin matrices mismatch
Text S3. Polynomial regression
Text S4. Introduction to Boston housing dataset
Text S5. Analysis of least squares
Text S6. Computing performance benchmarking
Fig. S1. Current-voltage characteristics of the Ti/HfO_2/C RRAM device.
Fig. S2. Cross-point resistive memory circuit on a printed circuit board.
Fig. S3. Device programming.
Fig. S4. Convergence analysis of the linear regression experiment.
Fig. S5. Extended circuit for one-step prediction.
Fig. S6. More linear regression results.
Fig. S7. Linear regression with two independent variables.
Fig. S8. Polynomial regression result.
Fig. S9. Logistic regression results.
Fig. S10. Convergence analysis of the logistic regression experiment.
Fig. S11. Solution of linear/logistic regression with negative independent variable values.
Fig. S12. Rescaling the attribute matrix $\boldsymbol{X}$ and price vector $\boldsymbol{y}$.
Fig. S13. One-step prediction circuit schematic for Boston housing dataset.
Fig. S14. Random first-layer weight matrix $\boldsymbol{W}^{(1)}$.
Fig. S15. The hidden-layer output matrix $\boldsymbol{X}$.
Fig. S16. Training and inference of the two-layer neural network.
Fig. S17. Linear regression of Boston housing dataset with a RRAM model.
Fig. S18. Impact of wire resistance.
Fig. S19. Linear regression of Boston housing dataset and its representative subsets.
Fig. S20. Scaling behavior of computing time of linear regression.
Fig. S21. Analysis of device variation impact and computing time.

REFERENCES AND NOTES

1. D. B. Strukov, G. S. Snider, D. R. Stewart, R. S. Williams, The missing memristor found. *Nature* **453**, 80–83 (2008).
2. D. Kau, S. Tang, I. V. Karpov, R. Dodge, B. Klehn, J. A. Kalb, J. Strand, A. Diaz, N. Leung, J. Wu, S. Lee, T. Langtry, Kuo-wei Chang, C. Papagianni, J. Lee, J. Hirst, S. Erra, E. Flores, N. Righos, H. Castro, G. Spadini, A stackable cross point phase change memory, in *Proceedings of the 2009 IEEE International Electron Devices Meeting (IEDM)* (2009), pp. 27.1.1–27.1.4.
3. M.-J. Lee, C. B. Lee, D. Lee, S. R. Lee, M. Chang, J. H. Hur, Y.-B. Kim, C.-J. Kim, D. H. Seo, S. Seo, U.-I. Chung, I.-K. Yoo, K. Kim, A fast, high-endurance and scalable non-volatile memory device made from asymmetric Ta_2O_{5-x}/TaO_{2-x} bilayer structures. *Nat. Mater.* **10**, 625–630 (2011).
4. T. Tuma, A. Pantazi, M. Le Gallo, A. Sebastian, E. Eleftheriou, Stochastic phase-change neurons. *Nat. Nanotechnol.* **11**, 693–699 (2016).
5. G. Pedretti, V. Milo, S. Ambrogio, R. Carboni, S. Bianchi, A. Calderoni, N. Ramaswamy, A. S. Spinelli, D. Ielmini, Memristive neural network for on-line learning and tracking with brain-inspired spike timing dependent plasticity. *Sci. Rep.* **7**, 5288 (2017).
6. S. Balatti, S. Ambrogio, Z. Wang, D. Ielmini, True random number generation by variability of resistive switching in oxide-based devices. *IEEE J. Emerging Topics in Circuits and Systems (JETCAS)* **5**, 214–221 (2015).
7. H. Jiang, D. Belkin, S. E. Savel'ev, S. Lin, Z. Wang, Y. Li, S. Joshi, R. Midya, C. Li, M. Rao, M. Barnell, Q. Wu, J. J. Yang, Q. Xia, A novel true random number generator based on a stochastic diffusive memristor. *Nat. Commun.* **8**, 882 (2017).
8. D. Ielmini, H.-S. P. Wong, In-memory computing with resistive switching devices. *Nat. Electron.* **1**, 333–343 (2018).
9. M. Cassinerio, N. Ciocchini, D. Ielmini, Logic computation in phase change materials by threshold and memory switching. *Adv. Mater.* **25**, 5975–5980 (2013).
10. Z. Sun, E. Ambrosi, A. Bricalli, D. Ielmini, Logic computing with stateful neural networks of resistive switches. *Adv. Mater.* **30**, 1802554 (2018).
11. G. W. Burr, R. M. Shelby, S. Sidler, C. di Nolfo, J. Jang, I. Boybat, R. S. Shenoy, P. Narayanan, K. Virwani, E. U. Giacometti, B. N. Kurdi, H. Hwang, Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element. *IEEE Trans. Electron Devices* **62**, 3498–3507 (2015).
12. T. Gokmen, Y. Vlasov, Acceleration of Deep Neural Network Training with Resistive Cross-Point Devices: Design Considerations. *Front. Neurosci.* **10**, 333 (2016).
13. P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, Y. Xie, PRIME: A Novel Processing-in-memory Architecture for Neural Network Computation in ReRAM-based Main Memory, in *Proceedings of the 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)* (2016), pp. 27–39.
14. S. Ambrogio, P. Narayanan, H. Tsai, R. M. Shelby, I. Boybat, C. di Nolfo, S. Sidler, M. Giordano, M. Bodini, N. C. P. Farinha, B. Killeen, C. Cheng, Y. Jaoudi, G. W. Burr, Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature* **558**, 60–67 (2018).
15. 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).
16. M. Le Gallo, A. Sebastian, R. Mathis, M. Manica, H. Giefers, T. Tuma, C. Bekas, A. Curioni, E. Eleftheriou, Mixed-precision in-memory computing. *Nat. Electron.* **1**, 246–253 (2018).
17. M. A. Zidan, Y. Jeong, J. Lee, B. Chen, S. Huang, M. J. Kushner, W. D. Lu, A general memristor-based partial differential equation solver. *Nat. Electron.* **1**, 411–420 (2018).
18. Z. Sun, G. Pedretti, E. Ambrosi, A. Bricalli, W. Wang, D. Ielmini, Solving matrix equations in one step with cross-point resistive arrays. *Proc. Natl. Acad. Sci. U.S.A.* **116**, 4123–4128 (2019).
19. M. H. Kutner, C. J. Nachtsheim, J. Neter, W. Li, *Applied Linear Statistical Models* (McGraw-Hill, ed. 5, 2004).
20. S. Weisberg, *Applied Linear Regression* (John Wiley & Sons, ed. 3, 2005).
21. I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning* (MIT Press, 2016).
22. D. W. Hosmer, S. Lemeshow, *Applied Logistic Regression* (John Wiley & Sons, Hoboken, ed. 2, 2000).
23. C. M. Bishop, *Pattern recognition and machine learning* (Springer, 2006).
24. R. Rojas, *Neural Networks: A Systematic Introduction* (Springer-Verlag, 1996).
25. The State of Data Science & Machine Learning (2017); www.kaggle.com/surveys/2017.
26. R. Penrose, A generalized inverse for matrices. *Math. Proc. Cambridge Philos. Soc.* **51**, 406–413 (1955).
27. E. Ambrosi, A. Bricalli, M. Laudato, D. Ielmini, Impact of oxide and electrode materials on the switching characteristics of oxide ReRAM devices. *Faraday Discuss.* **213**, 87–98 (2019).
28. E. M. Cherry, A new result in negative-feedback theory, and its application to audio power amplifiers. *Int. J. Circuit Theory Appl.* **6**, 265–288 (1978).
29. J. Fan, I. Gijbels, *Local Polynomial Modelling and Its Applications* (Chapman & Hall, 1996).
30. W. Ashton, *The Logit Transformation* (Charles Griffin & Co., 1972).
31. D. Harrison Jr., D. L. Rubinfeld, Hedonic prices and the demand for clean air. *J. Environ. Econ. Manag.* **5**, 81–102 (1978).
32. Boston Housing (2016); www.kaggle.com/c/boston-housing.
33. W. F. Schmidt, M. A. Kraaijveld, R. P. Duin, Feed Forward Neural Networks With Random Weights, in *Proceedings of the 11th IAPR International Conference on Pattern Recognition* (IEEE, 1992), pp. 1–4.
34. G. B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: Theory and applications. *Neurocomputing* **70**, 489–501 (2006).
35. D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).
36. Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998).
37. D. Yu, L. Deng, Efficient and effective algorithms for training single-hidden-layer neural networks. *Pattern Recognit. Lett.* **33**, 554–558 (2012).
38. D. C. Ciresan, U. Meier, J. Schmidhuber, Transfer Learning for Latin and Chinese Characters with Deep Neural Networks. *Proc. Int. Joint Conf. Neural Netw. (IJCNN)* **20**, 1–6 (2012).
39. K. Seo, I. Kim, S. Jung, M. Jo, S. Park, J. Park, J. Shin, K. P. Biju, J. Kong, K. Lee, B. Lee, H. Hwang, Analog memory and spike-timing-dependent plasticity characteristics

of a nanoscale titanium oxide bilayer resistive switching device. *Nanotechnology* **22**, 254023 (2011).
40. J. Park, M. Kwak, K. Moon, J. Woo, D. Lee, H. Hwang, TiO_x-Based RRAM Synapse with 64-levels of Conductance and Symmetric Conductance Change by Adopting a Hybrid Pulse Scheme for Neuromorphic Computing. *IEEE Electron Dev. Lett.* **37**, 1559–1562 (2016).
41. J. Tang, D. Bishop, S. Kim, M. Copel, T. Gokmen, T. Todorov, S. H. Shin, K.-T. Lee, P. Solomon, K. Chan, W. Haensch, J. Rozen, ECRAM as Scalable Synaptic Cell for High-Speed, Low-Power Neuromorphic Computing, in *Proceedings of the 2018 IEEE International Electron Devices Meeting (IEDM 2018)* (2018), pp. 13.1.1–13.1.4.
42. T.-C. Chang, K.-C. Chang, T.-M. Tsai, T.-J. Chu, S. M. Sze, Resistance random access memory. *Mater. Today* **19**, 254–264 (2016).
43. A. Mehonic, A. L. Shluger, D. Gao, I. Valov, E. Miranda, D. Ielmini, A. Bricalli, E. Ambrosi, C. Li, J. J. Yang, Q. Xia, A. J. Kenyon, Silicon Oxide (SiO_x): A Promising Material for Resistance Switching? *Adv. Mater.* **30**, 1801187 (2018).
44. International Technology Roadmap for Semiconductors (ITRS); http://www.itrs2.net/itrs-reports.html.
45. F. Tisseur, K. Meerbergen, The quadratic eigenvalue problem. *SIAM Rev.* **43**, 235–286 (2001).
46. K. Moon, A. Fumarola, S. Sidler, J. Jang, P. Narayanan, R. M. Shelby, G. W. Burr, H. Hwang, Bidirectional non-filamentary RRAM as an analog neuromorphic synapse, Part I: Al/Mo/$Pr_{0.7}Ca_{0.3}Mn_{O3}$Material improvements and device measurements. *IEEE J. Electron Dev. Soc.* **6**, 146–155 (2018).
47. C. S. Yang, D. S. Shang, N. Liu, G. Shi, X. Shen, R. C. Yu, Y. Q. Li, Y. Sun, A Synaptic Transistor based on Quasi-2D Molybdenum Oxide. *Adv. Mater.* **29**, 1700906 (2017).
48. E. J. Fuller, S. T. Keene, A. Melianas, Z. Wang, S. Agarwal, Y. Li, Y. Tuchman, C. D. James, M. J. Marinella, J. J. Yang, A. Salleo, A. A. Talin, Parallel programming of an ionic floating-gate memory array for scalable neuromorphic computing. *Science* **364**, 570–574 (2019).
49. N. P. Jouppi, N. Boden, A. Borchers, R. Boyle, Pierre-luc Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. Richard Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. M. Kean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, D. H. Yoon, in-Datacenter Performance Analysis of a Tensor Processing Unit, in *Proceedings of the 44th Annual International Symposium on Computer (ISCA'17)* (2017), pp. 1–12.
50. 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).

Acknowledgments: We thank J. Li for help with the SPICE simulation. **Funding:** This article has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement no. 648635). This work was partially performed at PoliFAB, the micro- and nanofabrication facility of Politecnico di Milano. **Author contributions:** Z.S. conceived the idea and designed the circuit. G.P. designed the printed circuit board. G.P. and Z.S. conducted the experiments. A.B. fabricated the devices. All the authors discussed the experimental and simulation results. Z.S. and D.I. wrote the manuscript with input from all authors. D.I. supervised the research. **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 the paper and/or the Supplementary Materials. Additional data related to this paper may be requested from the authors.

Submitted 31 May 2019
Accepted 21 November 2019
Published 31 January 2020
10.1126/sciadv.aay2378

Citation: Z. Sun, G. Pedretti, A. Bricalli, D. Ielmini, One-step regression and classification with cross-point resistive memory arrays. *Sci. Adv.* **6**, eaay2378 (2020).

One-step regression and classification with cross-point resistive memory arrays

Zhong Sun, Giacomo Pedretti, Alessandro Bricalli and Daniele Ielmini

Sci Adv **6** (5), eaay2378.
DOI: 10.1126/sciadv.aay2378