Improvement Learning Using a Fully Integrated Neuro-Inspired Memristor Chip

Huaqiang Wu (wuhq@tsinghua.edu.cn)
Tsinghua University https://orcid.org/0000-0001-8359-7997

Bin Gao
Tsinghua University https://orcid.org/0000-0002-2417-983X

Wenbin Zhang
Tsinghua University

Peng Yao
Tsinghua University

Qi Liu
Tsinghua University

Dong Wu
Tsinghua University

Qingtian Zhang
Institute of Microelectronics, Tsinghua University, Beijing, 100084, https://orcid.org/0000-0003-2732-3419

Minghong Xu
Tsinghua University

Ying Zhou
Tsinghua University

Zhenhua Zhu
Tsinghua University

Yi Cai
Tsinghua University

Dabin Wu
Tsinghua University

Jianshi Tang
School of Integrated Circuits, Tsinghua University

He Qian
Tsinghua University

Yu Wang
Tsinghua University

Physical Sciences - Article

Keywords:

Posted Date: May 18th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1606598/v1

Improvement Learning Using a Fully Integrated Neuro-Inspired Memristor Chip

Wenbin Zhang[1,3], Bin Gao[1,3,*], Peng Yao[1,3], Qi Liu[1], Dong Wu[1], Qingtian Zhang[1], Minghong Xu[1], Ying Zhou[1], Zhenhua Zhu[2], Yi Cai[2], Dabin Wu[1], Jianshi Tang[1], He Qian[1], Yu Wang[2], and Huaqiang Wu[1,*]

[1]School of Integrated Circuits, Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing, China.

[2]Department of Electronic Engineering, Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing, China.

[3]These authors contributed equally: Wenbin Zhang, Bin Gao, Peng Yao.

[*]e-mail: gaob1@tsinghua.edu.cn, wuhq@tsinghua.edu.cn

Abstract

Future neuro-inspired computing chips will feature high energy efficiency and extensive learning capabilities. Among these features, improvement learning is of great importance for edge intelligence devices to adapt to different application scenes and owners. However, current technologies for training neural networks greatly rely on large-scale computers, which are based on von Neumann architectures and complex digital circuits. The huge energy costs of moving massive data between computing and memory units and processing high-precision data hinder the implementation of improvement learning on

edge devices. Here, we report improvement learning with a low-power full-system-integrated memristor-based neuro-inspired computing chip. We propose a memristor-featured analogue in-memory computing architecture for implementing on-chip learning efficiently and integrate ~160K memristor cells with complete peripheral circuits in a complementary metal-oxide-semiconductor (CMOS)-compatible technology. We demonstrate the on-chip learning of both new samples and new classes across various tasks, including motion control, image classification, and speech recognition. The energy consumption of the memristor chip for learning is three orders of magnitudes lower than that of a central processing unit (CPU)-based system based on hardware system measurements. This study is an important step toward the learning chips with much higher energy efficiency, which will facilitate the development of future smart edge devices that can adapt to all kinds of special scenes.

Main

Human's ability of improvement learning, which features the fast learning of new knowledge while maintaining pre-acquired knowledge, plays a vital role in the growth of intelligence and the fast adaptation to dynamically changing environments. As illustrated in Fig. 1a, the learning of new knowledge (e.g., new samples or new classes) is quickly realized with only a few new inputs, and this is done without losing pre-acquired knowledge. Edge artificial intelligence (AI) applications also require hardware to have such improvement learning abilities to enable the associated devices to adapt to new scenes or user habits[1]. However, deep neural network (DNN) training[2,3], no matter improvement learning, transfer

learning, or learning from scratch, is typically implemented with conventional hardware based on a von Neumann computing architecture and a high-precision digital computing paradigm[4]. The extensive data movement between the processor chip and the off-chip main memory incurs massive energy consumption and accounts for most of the latency of the whole training process[5,6]. Therefore, although cloud computing platforms can handle such energy-intensive training[4,7], their high energy consumption hinders the implementation of improvement learning on power-limited edge computing platforms[1]. In contrast, memristor-based neuro-inspired computing eliminates this extensive data movement via its disruptive computation-in-memory architecture and analogue computing paradigm[6,8,9]. A memristor crossbar array can store an analogue synaptic weight and perform *in situ* vector-matrix multiplication (VMM) operations in parallel in a single time step by exploiting Ohm's law and Kirchhoff's law. A neuro-inspired computing chip that integrates multiple memristor crossbar arrays and CMOS circuits can easily implement DNN inference[10–13] and has great potential to handle fully on-chip learning without any assistance from off-chip memory[14–16]. The significant energy efficiency enhancement provided by memristor-based neuro-inspired computing makes this paradigm very promising for developing future chips that can enable low-power improvement learning devices.

Several studies[17–23] have demonstrated learning using memristor crossbar arrays for *in situ* weight tuning while utilizing software or external digital processors to implement the backpropagation (BP) algorithm[2]. However, it is still very challenging to realize a complete full-system-integrated memristor chip with strong learning ability and low energy costs. The key challenge lies in the inefficiency of mapping the BP algorithm to on-chip hardware. First,

an in-memory implementation of the BP algorithm requires costly conductance tuning operations with write verification because of device nonidealities, such as device variability and nonlinear conductance modulation[14,24–27]. Second, it is difficult to achieve efficient parallel conductance tuning with write verification[18–20,22], which makes on-chip learning more time-consuming and energy-consuming. Third, the high-precision data processing operations required during weight update calculations require a large circuit area and high energy consumption, leading to unacceptable overhead[24,28].

In this work, we demonstrate a memristor-based neuro-inspired computing chip that enables fully on-chip improvement learning, for which a memristor-featured **s**ign- and **t**hr**e**sho**l**d-based **l**e**ar**ning (STELLAR) architecture is proposed. In this architecture, first, the on-chip updating scheme is proposed to tune the memristor without verification. This scheme saves excessive write-and-read costs in the conductance tuning operations during the write verification scheme. Second, the on-chip calculation module is optimized to determine the weight update direction, and this process solely involves the signs of the inputs, outputs, and errors instead of their high-precision formats. This design reduces the circuit design burden and avoids massive overhead during on-chip learning. Third, a cycle-parallel conductance tuning scheme is proposed, in which conductance tuning is performed in a row-by-row parallel fashion. This scheme further reduces the induced energy consumption and latency and accommodates the limited endurance of memristors.

The fabricated neuro-inspired computing chip integrates two memristor crossbar arrays (~160K cells in total) and all the necessary circuit modules, including controllers for configuration, drivers for computing and programming, low-cost data converters, and

memristor-featured learning modules (as illustrated in Fig. 1b). Based on the obtained hardware-measured results, the memristor chip exhibits a reduction in energy consumption of more than three orders of magnitudes relative to that of the CPU-based system (an Intel i9-10900K). The learning of both the new samples and new classes are experimentally demonstrated across various tasks, including motion control for a light-chasing car, image classification, and audio recognition. The memristor-based neuro-inspired computing chip can facilitate the development of edge AI devices that can adapt to new scenes and new users, as illustrated in Fig. 1b.

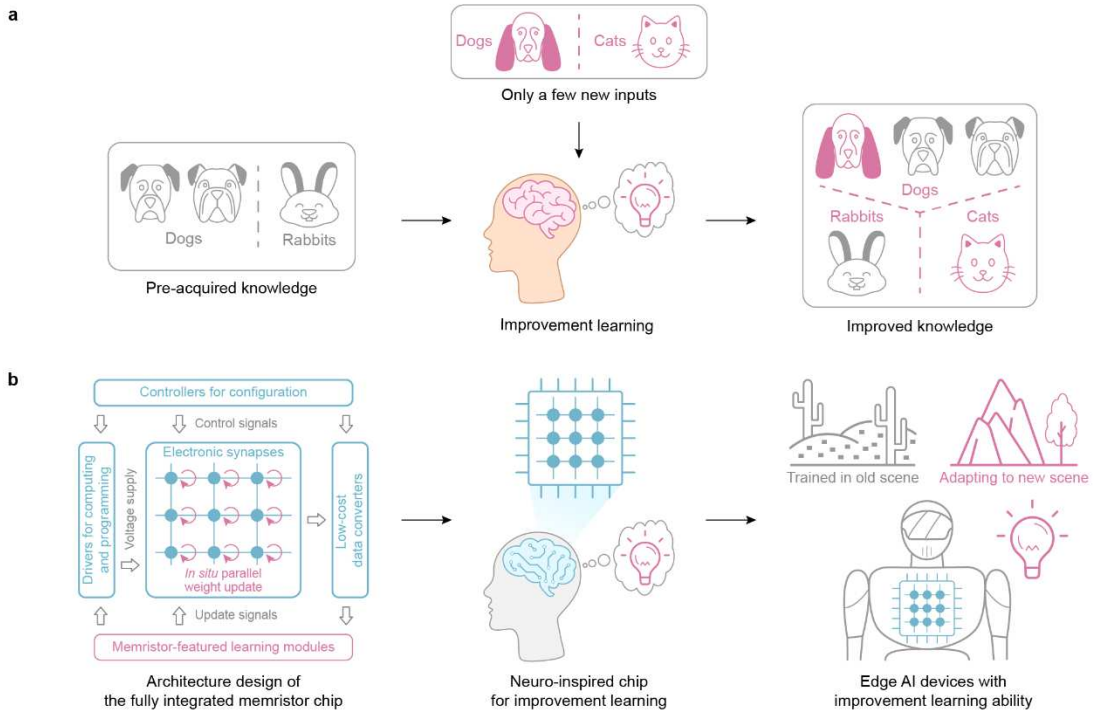


Fig. 1 | Improvement learning with a neuro-inspired memristor chip. a, Illustration of the improvement learning ability possessed by human brains. With pre-acquired knowledge about old dogs and rabbits, the learning of new samples (i.e., new dogs) and a new class (i.e., cats) is quickly realized with only a few new inputs. **b,** Design considerations and future applications of the memristor-based neuro-inspired computing chip. The chip, designed for fully on-chip learning, integrates all the necessary modules with memristor arrays. It equips edge AI devices with the improvement learning ability, enabling them to quickly adapt to

new scenes.

Memristor-featured architecture for on-chip learning

To support on-chip learning with appreciable energy efficiency, area efficiency, and accuracy, we propose the STELLAR architecture, as illustrated in Fig. 2a. The STELLAR architecture exploits the bidirectional analogue switching behaviour of the memristor device[29]. During the weight update stage, only the weight update direction needs to be calculated based on the signs of the inputs, outputs, and errors. In addition, the architecture predefines a threshold, which filters out the small errors when calculating the error signs and plays a vital role in the convergence of the learning algorithm by avoiding updates that are too sensitive and unnecessary. This threshold is hardware-reconfigurable to adapt to various learning tasks. The algorithmic details of the STELLAR architecture can be found in the Methods section (Algorithm for the STELLAR architecture). Depending on the weight update direction, a corresponding identical SET or RESET pulse is applied to the memristor cells. With this scheme, we can realize energy-efficient hardware by avoiding the complex precise weight update calculation and write verification processes and the complex peripheral circuit design.

The learning performance of the STELLAR architecture is compared with that of the conventional methods via simulations on the Modified National Institute of Standards and Technology[30] (MNIST) dataset. Here, all memristors are set to random conductance states before the learning process starts. Fig. 2b shows the learning accuracies of the conventional BP method without and with different write variations (1% and 3%, given as the percentage of the full conductance window of the memristor device) in comparison with those of the proposed

method under various thresholds. The simulation details can be found in the Methods section (Comparison between STELLAR and the conventional BP algorithm). A properly selected threshold yields improved convergence and learning accuracy, as shown in Extended Data Fig. 1a. A threshold that is too small leads to weight updates that are too frequent and an oscillating state of the network, while a threshold that is too large leads to inadequate weight updates. The comparison of the energy consumption among different methods is shown in Fig. 2c. While maintaining almost the same accuracy, the energy consumption of the STELLAR architecture is two orders of magnitudes lower than that of the conventional BP method due to the significant reductions in the precise weight update calculations and the write verification overhead.

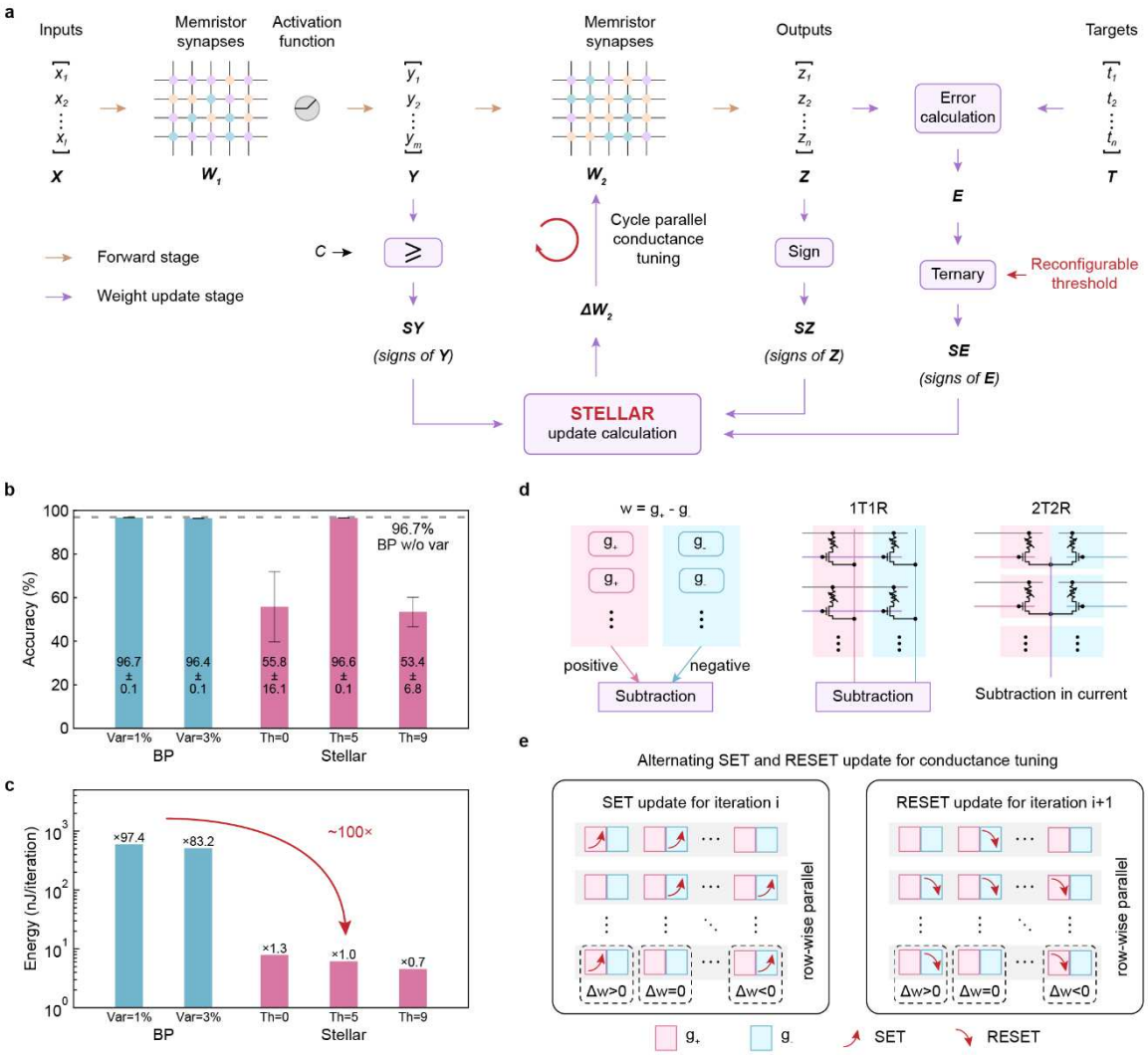


Fig. 2 | Design of the memristor-featured architecture for on-chip learning. a, Diagram of the STELLAR architecture used in the memristor chip. The STELLAR algorithm features a sign-based weight update calculation and a reconfigurable threshold for the error sign calculation during the weight update stage. **b**, **c**, Comparison of the classification accuracies **(b)** and the weight update energy consumptions **(c)** using the STELLAR and the conventional BP algorithm on the MNIST dataset in simulations. **b** and **c** show the results of different write variations when utilizing the BP algorithm and the results of different thresholds in the STELLAR learning approach. The bars and error bars show the averages and standard deviations of the results of 10 repeated experiments, respectively. The dashed line in **b** shows the average accuracy obtained by BP without considering the write variations of the memristors. **d**, Representations of weights with a differential conductance pair (left) and with 1T1R (center) and 2T2R (right) configurations. **e**, Illustration of the cycle-parallel conductance tuning scheme. The red and blue rectangles represent the positive and negative memristor cells in each differential pair, respectively. The upwards and downwards arrows represent

the SET and RESET operations on the associated memristor cell, respectively.

The STELLAR architecture realizes positive and negative weights with differential pairs of memristor cells[19,22,31,32], as illustrated in Fig. 2d. In conventional crossbar arrays with one-transistor-one-resistor (1T1R) configurations, the two memristor cells in a differential pair are connected to different source lines (SLs), and subtraction is accomplished in a digital fashion. In crossbar arrays with two-transistor-two-resistor (2T2R) configurations, the two memristor cells in a differential pair are connected to the same SL, and subtraction is accomplished directly in the current domain[10]. The 2T2R design greatly reduces the SL current and thus reduces IR drop issues, thus enabling a larger array size. Here, we propose a cycle-parallel conductance tuning scheme for such a differential pair configuration, as illustrated in Fig. 2e. In this scheme, the SET and RESET operations are performed alternatively for the learning iterations of arriving input samples (e.g., images). If the SET (RESET) operation is performed in the current learning iteration, then the RESET (SET) operation is performed in the next learning iteration. Only one operation is performed on the whole array in each iteration; this is achieved by selecting which cell of a given differential pair to operate on. Take the SET update mode as an example, if the weight update direction is positive (i.e., $\Delta W>0$), the positive cell is updated with SET operation to increase the weight; if the $\Delta W<0$, the negative cell is updated with SET operation to decease the weight; and if $\Delta W=0$, neither the positive cell nor negative is updated. The conductance tuning is performed in a row-by-row parallel fashion, and the detailed schematics of the array operations are shown in Extended Data Fig. 2. The memristor devices in the same row are selected through their word line (WL) signals and tuned depending

on the corresponding SL signals, while the bit line (BL) signals remain constant. The cycle-parallel conductance tuning scheme can be applied to either 1T1R or 2T2R memristor arrays. Because only one-half of the memristor devices are updated during each on-chip learning iteration, the cycle-parallel conductance tuning scheme reduces the induced energy consumption and alleviates the requirement regarding the memristor endurance.

Chip design, fabrication, and measurements

Fig. 3a shows the overall circuit implementation of the proposed STELLAR architecture. This memristor chip consists of controllers for configuration; a 2T2R memristor array (1568 × 100), a 1T1R memristor array (100 × 20); BL, WL, and SL drivers for computing and programming; low-cost analogue-to-digital converters (ADCs); modules for memristor-featured on-chip learning (i.e., error circulating subtractors and weight update logic); and input and output buffers. The 1st-layer memristor array adopts a 2T2R configuration to reduce the IR drop issues that occur in such a large array, while the 2nd-layer memristor array adopts a 1T1R configuration to support more flexible *in situ* weight tuning. The controllers decode the input stage selection signals and provide the output configuration signals to other circuit modules to switch the chips to different working stages (see Methods: Circuit design of the memristor chip and Extended Data Fig. 3). The resolutions of the ADCs are adjustable, called AR-ADC, allowing for flexible threshold reconfiguration[10]. The error calculation is accomplished with subtractors, which are realized with counters. The weight update logic determines the weight update direction and the conductance tuning operations.

A micrograph of the fabricated chip is shown in Fig. 3b. The chip area breakdown is

described in Extended Data Fig. 4b. The memristor device uses a material stack of TiN/HfO_x/TaO_y/TiN, and the fabrication process is compatible with the standard CMOS process (see Methods: Fabrication of the memristor chip and Extended Data Fig. 4a). As a result, the memristors can be conveniently integrated with complex CMOS circuits to produce an excellent yield (almost 100% of all 160K cells). The cross-section transmission electron microscopy (TEM) image in Fig. 3c shows the integration of memristor cells with CMOS circuity. The fabricated memristors exhibit uniform and repeatable bidirectional analogue switching with identical pulse trains, as shown in Extended Data Fig. 4d. The ~160K total on-chip memristor cells can be uniformly programmed to 32 conductance states, with maximum, minimum, and average success rates of 99.98%, 99.69%, and 99.90%, respectively (see Methods: Measurements of the memristor devices and Extended Data Fig. 4c).

The on-chip inference is first demonstrated for MNIST handwritten digit classification. The weights are off-chip trained and then transferred to the chip as memristor conductance (see Methods: Off-chip training and on-chip inference). The measured classification accuracy of each class (0~9) is presented in Fig. 3d. The average accuracy is 95.8%. The effect of cell conductance fluctuations on the chip accuracy is also evaluated, as shown in Fig. 3e. We monitor the accuracy for 48 days, and no obvious accuracy degradation is observed. We also demonstrate real-time handwritten digit recognition with the memristor chip (see Supplementary Video 3).

An on-chip learning task, MNIST image classification, is further demonstrated to verify the on-chip learning ability based on a 784-100-10 multilayer perceptron (MLP). The weights in the 1st layer are off-chip trained and then transferred to the chip as memristor conductance.

The memristors in the 2nd layer are firstly programmed to the high-resistance state (HRS) and then updated using the STELLAR scheme. All data processing and signal control processes are executed on the chip. After three epochs of on-chip learning with the training set, the classification accuracies are increased from 8.6% and 8.4% to 94.9% and 92.3% on the training set and test set, respectively, as shown in Fig. 3f. We then evaluate the energy consumption of the on-chip learning with hardware-measured results (see Methods: Energy consumption benchmark). Fig. 3g shows the comparison of energy consumption between our memristor chip and the CPU-based system. The energy consumption is reduced by more than three orders of magnitudes (>2800 μJ versus 1 μJ) for one learning iteration with the same MLP network. We also evaluate the energy consumption of a graphics processing unit (GPU)-based system for one training iteration with the same MLP network. The energy reduction ratio reaches 86. The energy breakdown of the memristor chip during the on-chip learning process is presented in Fig. 3h. The energy consumption can be further reduced by optimizing the ADC design.[33–35]

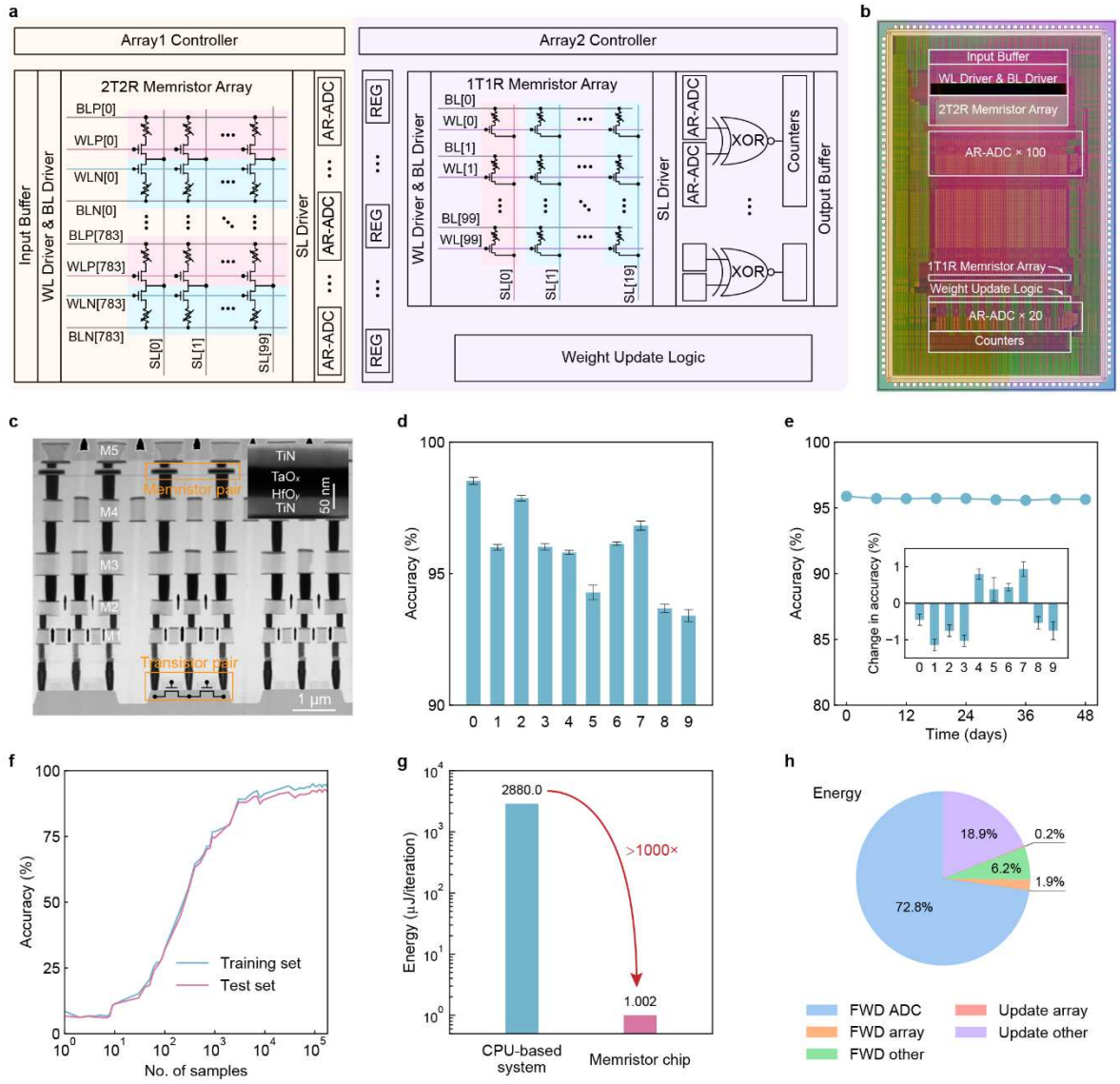


Fig. 3 | The memristor chip for on-chip learning. a, Overview of the architecture of the memristor chip. **b**, Optical microscope image of the chip, where several key components are labelled. **c**, A cross-section TEM image showing 2T2R cells. The transistors in a 2T2R cell share a common source terminal. Inset: cross-section TEM image of the memristor device. **d**, On-chip classification accuracy obtained for each class by using the off-chip trained weights for the MNIST dataset, with bars and error bars showing the averages and standard deviations of the accuracies achieved over 5 inference iterations. **e**, Changes in the classification accuracy over 48 days after weight transfer, with each point showing the average accuracy over 5 inference iterations. Inset: the average changes in the accuracy of each class after 48 days of weight transfer. **f**, Changes in the classification accuracy over 3 learning epochs in the on-chip learning task for the MNIST dataset. **g**, Comparison of the measured energy consumption between the CPU-based system and the memristor chip. **h**, Energy breakdown of the memristor chip during the on-chip learning process.

On-chip improvement learning

The memristor chip is used to further demonstrate four improvement learning tasks, including the motion control task of learning new samples, the audio recognition task of learning new samples, the image classification task of learning a new class, and the motion control task of learning a new class. The implementation of improvement learning mainly includes two stages. First, a model is off-chip trained on the base dataset, and then the model is transferred to the chip. Next, on-chip improvement learning with new data is performed based on the transferred model.

We first demonstrate the learning of new samples in a motion control task of a light-chasing car, as illustrated in Fig. 4a. The car is designed to pursue the location of a laser light spot; it is equipped with a camera to capture environmental images, a steering motor for direction control, and a driving motor for throttle control. The memristor chip receives the input features of the environmental images from the PC and provides the output control signals for the steering angles and driving throttles (see Methods: Motion control task of learning new samples for details). As illustrated in Fig. 4b, a convolutional neural network (CNN) including two fully connected (FC) layers with dimensions of 512×100×10 is first off-chip trained with old scene data (i.e., dark scene data), and then the weights of these two FC layers are transferred to the two corresponding layers of the memristor chip. Next, improvement learning of the new scene (i.e., a bright scene) is performed on the chip by tuning the weights of the last FC layer. Before improvement learning, the car sometimes loses track of the target (i.e., light spot) in the bright scene: it deviates from the target or moves forward even if no target is present. After

improvement learning, the car adapts well to the bright scene and still performs well in the dark scene (see Supplementary Video 1). Fig. 4c shows the evolution of the scores during improvement learning, where a score of 1 denotes the best performance (see Methods: Motion control task of learning new samples). The scores become stable after improvement learning with 500 training samples from the new scene. After improvement learning with all the training samples, the average score in the new scene significantly increases from 0.605 to 0.912, while that of the old scene increases from 0.951 to 0.963, showing that no degradation occurs, as presented in Fig. 4d. Fig. 4e shows the transitions and changes of conductance weights before and after the improvement learning.

We next demonstrate the learning of a new class in an image classification task involving the MNIST dataset, as illustrated in Fig. 4f. The base model is trained to recognize images of the digits 0 and 2-9 (old classes) and then transferred to the memristor chip. Next, the improvement learning of the new class (i.e., the digit 1) is performed on the chip. As shown in Fig. 4g, the accuracy achieved for the new class increases dramatically during the improvement learning with only a few training samples, while the accuracy achieved for the nine old classes is not significantly reduced. The accuracies yielded for the new class and old classes stabilize after 100 training samples. After improvement learning with 100 training samples, the average accuracy for the new class increases from 7.02% to 93.0%, while that achieved for the old classes decreases from 95.3% to 93.2%, as shown in Fig. 4h. The conductance weight distributions before and after improvement learning and their difference are shown in Fig. 4i.

ss

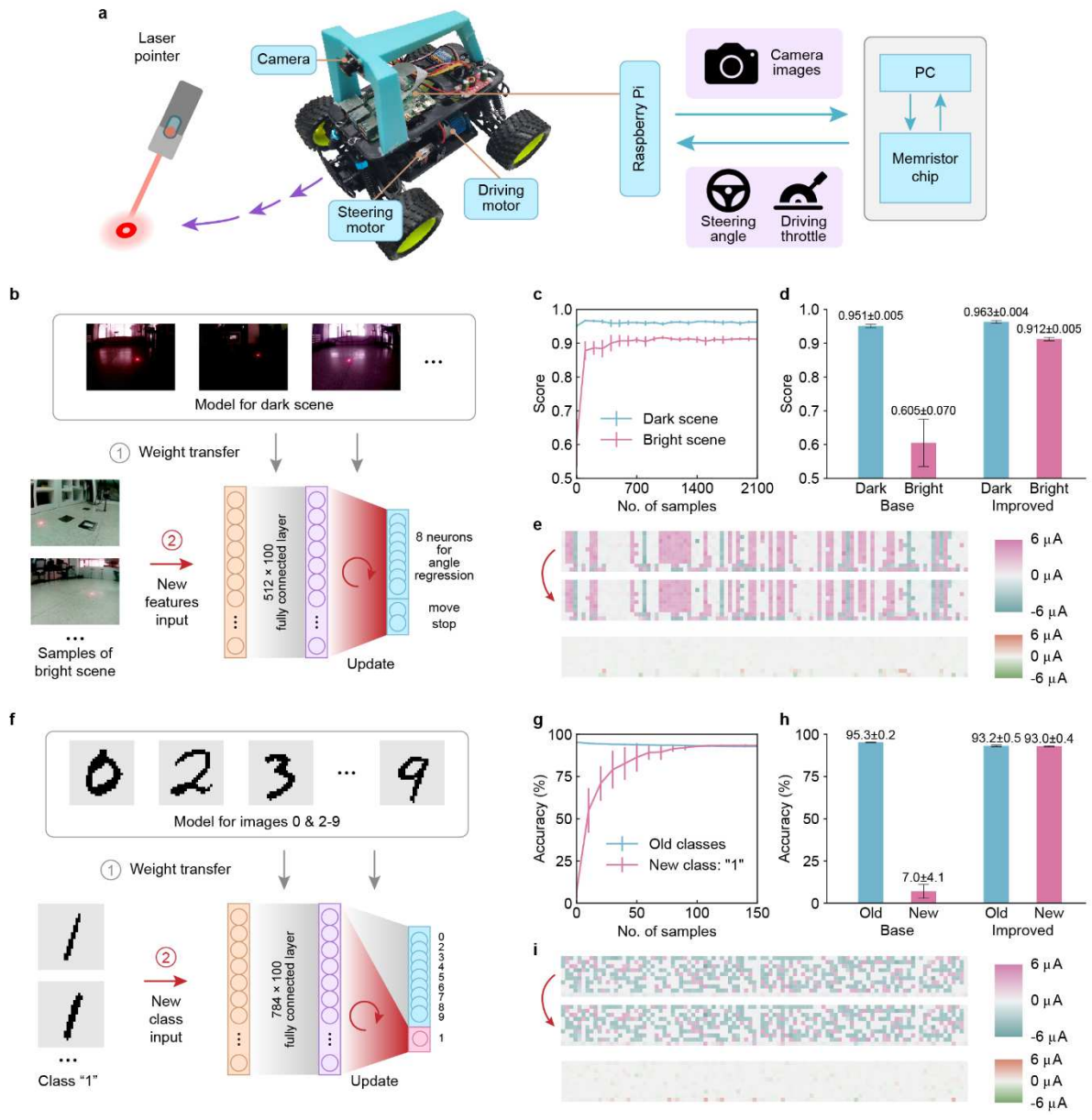


Fig. 4 | Improvement learning demonstrations on the memristor chip. a, Illustration of the motion control task (left) and diagram of its control system (right). **b**, Illustration of the learning of new samples in the light-chasing car task. **c**, Changes in the scores that the car produced in the dark and bright scenes over the course of on-chip improvement learning of bright scene samples. The lines and the error bars show the averages and the standard deviations of the scores of 10 repeated experiments with shuffled training sets, respectively. **d**, The initial and final scores in **c**. **e**, The evolution of the weight distribution (100 × 10) before (top panel) and after (middle panel) the on-chip improvement learning of bright scene samples. Bottom: distribution of the weight changes during improvement learning. **f**, Illustration of the learning of a new class in the image classification task. **g**, Changes in the classification accuracies achieved for the old classes and the new class

over the course of on-chip improvement learning of the new class. The lines and the error bars show the averages and the standard deviations of the accuracies of 10 repeated experiments with 150 randomly selected images from training sets, respectively. **h**, The initial accuracies and the accuracies achieved after on-chip improvement learning with 100 samples of the digit 1 in **g**. **i**, The evolution of the weight distribution (100 × 10) before (top panel) and after (middle panel) the on-chip improvement learning of the digit 1. Bottom: distribution of the weight changes during improvement learning.

In addition, the memristor chip is also used to implement an audio recognition task of learning new samples and a motion control task of learning a new class. In the audio recognition task, the on-chip improvement learning of female audio samples relies on the model pretrained with male audio samples, as illustrated in Extended Data Fig. 6a. In the motion control task of learning a new class, the on-chip improvement learning of the "moving backwards" action relies on the model knowing when to move forward or stop, as shown in Extended Data Fig. 7a. After improvement learning, the accuracy achieved for the new class (i.e., moving backwards) increases from 0% to 95.2%, while that for the old classes (i.e., moving forward or stopping) decreases from 89.5% to 89.4%. The details of the implementation of these two tasks can be found in the Methods Section (Audio recognition task of learning new samples; Motion control task of learning a new class).

In summary, we develop a full-system-integrated memristor chip with the improvement learning ability and a low energy cost. The schemes in the STELLAR architecture, including its learning algorithm, hardware realization, and parallel conductance tuning scheme, are general approaches that facilitate on-chip learning by using a memristor crossbar array, regardless of the type of memristor device. We demonstrate the improvement learning of both new samples and a new class across various tasks, including motion control, image

classification, and speech recognition, which shows that the STELLAR architecture accommodates the device nonidealities and equips the memristor chip with improvement learning ability to adapt to new scenarios. Furthermore, the benchmark involving our memristor chip shows that it requires three orders of magnitudes less energy consumption for on-chip learning than the CPU-based system. This study is an important step toward future chips with high energy efficiency and extensive learning capabilities. We hope our findings will accelerate the development of future smart edge devices that can adapt to different application scenarios and owners.

Methods

Algorithm for the STELLAR architecture

The STELLAR algorithm is developed to implement on-chip learning with memristors. The algorithm features sign-based weight update calculations and a reconfigurable threshold when calculating the signs of errors. The STELLAR algorithm is a generic approach that accelerates the implementation of on-chip learning with memristors for various neural networks. As the fabricated memristor chip realizes a two-layer neural network, the following description of the algorithm is based on this network. The details of the algorithm for other neural networks can simply be generalized from the following description.

A two-layer neural network with dimensions of 784 × 100 × 10 is constructed; it employs a rectified linear unit (ReLU) as an activation function for low-cost hardware implementation. Each weight value is represented by the differential conductance of a pair of memristors. The on-chip learning is composed of two stages: the forward stage and the weight update stage (as

shown in Extended Data Fig. 1c). The forward inference is performed with the memristor crossbar arrays (as illustrated in Fig. 2a):

$$\boldsymbol{Y} = ReLU(\boldsymbol{W_1^T} \cdot \boldsymbol{X})$$

$$\boldsymbol{Z} = \boldsymbol{W_2^T} \cdot \boldsymbol{Y}$$

where $\boldsymbol{X}$ is the input vector, $\boldsymbol{Y}$ is the 1st-layer output vector, $\boldsymbol{Z}$ is the 2nd-layer output vector, and $\boldsymbol{W_1}$ and $\boldsymbol{W_2}$ are the conductance weight matrices of the 1st layer and 2nd layer, respectively. The weight update is performed for each input vector. The learning algorithm aims to update the conductance properly to minimize the loss function of the network output and the target. The square loss function is used in this work, so the error vector is calculated by

$$\boldsymbol{E} = \boldsymbol{T} - \boldsymbol{Z}$$

where $\boldsymbol{T}$ is the target vector and $\boldsymbol{E}$ is the error vector. The signs of the errors are extracted to determine the weight update direction instead of the accurate values:

$$\boldsymbol{\Delta W_2} = \boldsymbol{SY} \cdot (\boldsymbol{SZ^T} \odot \boldsymbol{SE^T})$$

where $\boldsymbol{SY}$ is the sign vector of the 1st-layer output, $\boldsymbol{SZ}$ is the sign vector of the 2nd-layer output, and $\boldsymbol{SE}$ is the ternary sign vector of the error. The elements *sy*, *sz*, and *se* in the sign vectors $\boldsymbol{SY}$, $\boldsymbol{SZ}$, and $\boldsymbol{SE}$ are defined as follows:

$$sy = \begin{cases} 1, & y \geq C \\ 0, & y < C \end{cases}$$

$$sz = sign(z)$$

$$se = \begin{cases} +1, & e \geq Th \\ -1, & e \leq -Th \\ 0, & else \end{cases}$$

where *y*, *z*, and *e*, are the elements of vectors $\boldsymbol{Y}$, $\boldsymbol{Z}$, and $\boldsymbol{E}$, respectively. The *C* in the calculation of $\boldsymbol{SY}$ can be flexibly reconfigured, and its typical value is the maximum among the 1st-layer outputs multiplied by 0.4. The threshold *Th* filters out the small error values, i.e., $|se|<Th$,

which prevents weight updates that are too sensitive and improves the convergence of the algorithm.

Comparison between STELLAR and the conventional BP algorithm

We evaluate the performance of STELLAR and the conventional BP method in an on-chip learning simulation with memristors. In the STELLAR architecture, the weight update calculation is accomplished by custom-designed circuits, and the conductance tuning operations are accomplished without verification. In the BP method, the weight update calculation is accomplished by an Intel Xeon E5-2699 processor, and the conductance tuning operations are accomplished by using a write-verify scheme. We estimate the accuracies and energy consumptions yielded by using the same training task for MNIST image classification. An MLP with dimensions of 784×100×10 is utilized when comparing these methods. We use a behavioural memristor device model in the simulation of conductance tuning operation, as it provides a final conductance distribution after executing a SET/RESET pulse on a given initial conductance[28]. In the BP method, the memristor is programmed to a certain margin with the target conductance as the margin middle value and a given variation as the margin upper/lower limit. The value of the variation is constant for all the conductance states and is defined as a certain percentage of the full conductance window.

We focus on the energy consumption of the weight update stage. The energy consumption required by the STELLAR architecture involves a weight update calculation utilizing custom-designed circuits and a one-time conductance tuning operation without verification. The energy consumption required by the BP method involves a weight update

calculation utilizing the CPU and multiple conductance tuning operations with verification, requiring the ADC to verify the device conductance during the tuning process. We combine the experimental data and simulation data to estimate the energy consumption. The energy consumption of the custom-designed circuits is obtained from the corresponding circuits used in our memristor chip with the Cadence simulator. The energy consumption of the Intel Xeon E5-2699 processor is estimated as the number of calculation operations divided by its energy efficiency[36]. The energy consumption of the conductance tuning operations is estimated as the number of tuning operations multiplied by the average energy of each operation. The number of tuning operations is obtained from the on-chip learning simulation. The average energy of a write operation is estimated from the measured results yielded by our memristor chip, while the energy of a read operation is obtained from a 130-nm ADC with an 8-bit resolution[37].

Circuit design of the memristor chip

The STELLAR algorithm is modified for its chip implementation, and the schematic is shown in Extended Data Fig. 1b. The selection of threshold is realized by configuring the output scales and target values. It is predefined to take the first 8 bits of 17-bit signed errors $\boldsymbol{E}$ when calculating the ternary signs $\boldsymbol{SE}$. As a result, the selection of the threshold is equivalent to adjusting the error scale, which is accomplished by adjusting the output scales and target values. The output scales of $\boldsymbol{Y}$ and $\boldsymbol{Z}$ can be modulated with the resolution-adjustable ADC, whose quantization scale is determined by its resolution.

The on-chip learning of the memristor chip involves three working stages: Forward (FWD), SET, and RESET (as shown in Extended Data Fig. 1c). The controllers decode the

input stage selection signals to the output signals for voltage selection. The voltage selection signals are taken as the control signals of the BL/WL and SL drivers; the voltage to be applied to the memristor array is selected through the multiplexers in the drivers.

In the FWD stage, VMM operations are performed with the memristor arrays and on-chip ADCs. The 1^{st}-layer array utilizes the 2T2R configuration to reduce the IR drop. The 2^{nd}-layer array utilizes the 1T1R configuration with the two adjacent columns representing positive and negative weights, respectively. The resolutions of the ADCs are adjustable, enabling the flexible adjustment of the output scale. A timing-dependent scheme based on the number of pulses is used in this memristor chip. If the resolutions of the 1^{st}-layer ADCs are configured as N bits, they generate 2^N pulses. For each pulse cycle of the 1^{st}-layer output, the 2^{nd}-layer performs one VMM. The outputs of the 1^{st}-layer ADC are stored in 1-bit registers and sampled simultaneously by the next memristor array as inputs. The subtraction of the differential column pair is accomplished with the logics connected to their associated ADCs, and the subtraction results are sampled simultaneously by counters and stored in output buffers. The error calculation between the output and target is accomplished simultaneously with another set of counters, which use the preloaded targets as their initial values.

The signs of the 1^{st}-layer outputs ($\boldsymbol{SY}$) and the results of comparing the 1^{st}-layer outputs ($\boldsymbol{Y}$) with the reconfigurable value C are also obtained in the FWD stage. Since the memristor chip utilizes a pulse-based coding scheme, this function can be realized with an AND logic gate, which takes the 1^{st}-layer outputs and the reconfigurable value as inputs. The reconfigurable value signal is high during the C^{th} cycle of the 1^{st}-layer output. In this cycle, the AND logic generates the sign signals, which are stored in the registers.

The SET and RESET stages involve determining the weight update direction and performing conductance tuning operations. The conductance tuning is performed in a row-by-row fashion, with a set of shift registers selecting which row to update. The calculation of the weight update direction involves the signs of the 1^{st}-layer outputs, 2^{nd}-layer outputs, and errors, where the first two groups of signs are obtained in the FWD stage and stored in registers. The first 8 bits with the highest significance of error are considered when calculating the weight update direction via combination logic.

A 2T2R memristor array is proposed to reduce the IR drop by decreasing the SL output current. Voltages with opposite polarities are applied to the two memristors in the differential pair during inference, where the conductance of these two memristors represent positive and negative weights, respectively. The current through the positive weights and negative weights connected to the same SL can be locally cancelled out. This direct subtract in current reduces the IR drop, which decreases the power consumption and boosts the computing parallelism.

The resolution-adjustable ADC consists of three submodules: an integrator, a comparator, and a shared digital-to-analogue converters (DAC) for each layer. First, the integrator converts the SL current to a voltage. Then, the DAC generates a voltage that increases with the clock. Simultaneously, the comparator compares the output voltages of the integrator and DAC and generates a timing-dependent quantized output based on the number of pulses. The resolution of the ADC is configured by setting the clock frequency of the DAC and the sampling frequency. The resolution-adjustable ADC enables a flexible trade-off between system accuracy and power consumption.

Fabrication of the memristor chip

The memristor arrays are monolithically integrated with CMOS peripheral circuits on a chip. Each memristor device is connected in series with a selector transistor, whose gate is taken as the effective control terminal during programming. As shown in Fig. 3c, the memristor device is sandwiched between metal-4 and metal-5 interconnects with a material stack of TiN/HfO_x/TaO_y/TiN.

The selector transistor, peripheral circuits, and bottom four metal interconnections are fabricated in a standard CMOS foundry by using a 130-nm process. The memristors and top metal interconnections are fabricated in a laboratory with a back-end-of-line process. First, the TiN bottom electrode layer, HfO_x switching layer, TaO_y capping layer, and TiN top electrode layer are sequentially deposited on the wafers from the foundry. The capping layer serves as a thermally enhanced layer to modulate the distributions of the electric field and temperature in the switching layer and contributes to the improved analogue switching behaviour. The deposited layers are patterned with lithography and etching to form isolated 0.7 μm × 0.7 μm memristor cells. Afterwards, a SiO_2 dielectric is deposited, polished, patterned, and etched, forming vias for interconnects. Finally, the top vias are formed by sputtering tungsten and conducting chemical mechanical polishing, and the top metal interconnections are formed by sputtering aluminium and shaping it, which completes the fabrication process.

Measurements of the memristor devices

The test system for the memristor chip mainly integrates a field-programmable gate array (FPGA) and relevant voltage generators. The FPGA generates control commands for, sends

inputs to, and reads results from the memristor chip. The voltage generators offer varying voltages for the VMM operation and memristor programming. The FPGA communicates with the host PC that realizes the user interface through an Ethernet connection. To facilitate the implementation of various learning tasks on the memristor chip, we map the high-level functions on PC, provided as application programming interfaces (APIs), to the fundamental operations of the memristor chip. The APIs include various programming schemes, inference, or learning on a given dataset.

The memristor device has a high yield for 5-bit (32-level) programming, with a minimum success rate of 99.69% across all the programmed conductance states (see Extended Data Fig. 4c). During the test, ~160K memristor devices are programmed to 32 target conductance states, with ~5,000 devices assigned to each state. The conductance targets range within a switching window from 2 μS to 20 μS with a uniform interval of 0.58 μS. The margin is set as ±0.24 μS for each target conductance state. The memristor conductance is sensed with on-chip ADCs at a 0.2 V read voltage after each programming pulse. To accelerate the high-precision multibit programming of memristor conductance, we develop a row-by-row program verification scheme in a parallel fashion[38]. This parallel programming scheme utilizes the incremental gate voltage programming method, which solely varies the gate voltage of the selector transistor during the conductance turning process. The maximum number of programming pulses is set to 130 when 10 cells are programmed in parallel; that is, 13 pulses are required per cell on average for 5-bit programming in the worst case.

Another 1K 1T1R memristor array on the same wafer with the neuro-inspired computing chip is used to characterize the analogue switching behaviour of the memristor device. Sixty-

four memristor cells are measured for 5 cycles consisting of alternated potentiation and depression. Potentiation refers to the evolution of conductance under an identical SET pulse train, which contains 128 SET pulses with 1.5 V and 50-ns pulse widths. Depression refers to the evolution of conductance under an identical RESET pulse train, which contains 128 RESET pulses with 1.4 V and 50-ns pulse widths.

Off-chip training and on-chip inference

The neural networks used in the following tasks are quantized to ternary weights (+1, 0, and −1) during off-chip training[39]. We inject noise into the weights during off-chip training to improve the resilience of the network against nonidealities[40]. The noise level here is defined as the ratio of the standard deviation of the programming error to the target conductance value. Based on-chip characterization involving single-side verification, the noise level is approximately 30%.

We first train the models with different levels of noise ranging from 0% to 50% and then select the model with the best inference accuracy at the 30% noise level. Extended Data Fig. 1a shows that the highest accuracy is achieved by the model with comparable noise injection during off-chip training. The weights of the off-chip-trained model are transferred to the chip as the memristor conductance during the weight transfer stage. Binarized images with resolutions of 28 × 28 pixels obtained from the MNIST dataset are used as the input images. The inference is performed on 10,000 images derived from the entire test set. We investigate the change in accuracy with time by using the same model. The inference is performed 5 times each day on the entire test set within 48 days after weight transfer.

Implementation of the on-chip learning task

We also use the MNIST dataset to demonstrate the on-chip learning task. The 1st-layer weights of the off-chip-trained model were transferred to the 1st-layer memristor array as conductance. The 2nd-layer memristors are all programmed to the HRS via single-side verification, and the programming noise works as the random initialization of the 2nd-layer weights. The on-chip learning is performed on 180,000 images (3 epochs with the entire training set containing 60,000 images). The inference is performed on 10,000 randomly selected images from the training set and the 10,000 images forming the entire test set.

Energy consumption benchmark

We measure the latency and the power consumption of the memristor chip in the FWD, SET, and RESET stages. The total energy consumption for one learning iteration is the sum of the energy consumption of the forward (FWD) stage and weight update stage, where the energy consumption of the weight update stage is the average energy consumption of the SET and RESET stages. It should be mentioned that the energy consumed by the ADCs in the SET and RESET stages is not considered because the ADCs can be powered off during these stages with a good chip design. From the above measurements and calculations, we obtain that the energy consumption of our memristor chip for each on-chip learning iteration is 1.002 μJ. The detailed metrics, including the latency, power, and energy consumption of each stage in the learning mode, are listed in Extended Data Table 1. The SET pulse width is 200 ns, while the RESET pulse width is 50 ns. As a result, the delays of the SET stage and RESET stage are different.

The energy consumption of the CPU-based system (Intel i9-10900K) for one training iteration is calculated as follows: thermal design power (125 W) × CPU usage during training (50.0%) × run time for one training iteration (46.08 μs) = 2880.0 μJ.

We also estimate the energy consumption of the GPU (NVIDIA K80)-based system for one training iteration. The operations required for conducting one training iteration with the GPU-based system include 1) loading all the weights from dynamic random-access memory (DRAM) to the GPU, 2) performing forward and weight update-related operations with the GPU, and 3) writing the 2nd-layer weights back to the DRAM. We assume floating-point (32-bit) weights, which are usually required for neural network training. The energy consumption of the GPU is estimated as the number of calculation operations divided by its energy efficiency[36]. The energy consumption of the DRAM is estimated as the number of writing/reading bits multiplied by the energy consumed by each bit, which includes the DRAM writing/reading energy and the on-board interconnection energy. The DRAM writing/reading energy is assumed to be 20 pJ per bit based on the reported data for Graphics Double Data Rate, version 5[41] (GDDR5), which is adopted in the NVIDIA K80 GPU. The on-board interconnection energy is assumed to be 10 pJ per bit[42]. The estimated energy consumed by the GPU, DRAM, and GPU-based system is 8.67 μJ, 77.18 μJ, and 85.85 μJ, respectively.

Motion control task of learning new samples

We demonstrate the improvement learning of new samples in the motion control task of a light-chasing car, whose control algorithm involves mapping a camera image to a control decision (i.e., a steering angle and a driving throttle) directly in an end-to-end learning

approach[43]. As illustrated in Extended Data Fig. 5a, a traditional approach is developed to generate control decisions for teaching neural networks and evaluating their performance. The traditional approach divides the control task into two parts, including laser spot detection and control logic generation. The camera images and corresponding control decisions are collected as datasets when the traditional approach drives the car. An end-to-end neural network is then trained with the collected samples to directly turn the images into control decisions. The obtained network is deployed on the car to drive it autonomously in real environments. In this task, the steering angles are analogue signals that range from -1 to 1, corresponding to different turning angles of the steering motor from the leftmost to the rightmost. The driving throttle is a binary signal corresponding to the driving motor moving forward or stopping.

A multitask CNN including convolutional layers and two FC layers is constructed to realize the control of the light-chasing car. The dimensions of the last two FC layers are 512 × 100 × 10 to fit the array size of the fabricated memristor chip. Eight of the ten outputs predict a regression value for the continuous steering angle, while the other two outputs classify the driving throttle, indicating whether to move or stop. The FC layers are implemented on the chip, while the convolutional layers are implemented with software. The on-chip learning is performed on 2,100 samples collected from the bright scene. The inference is performed on 4,502 samples acquired from the dark scene and 900 samples collected from the bright scene.

A customized scoring mechanism is developed to evaluate the performance of the neural network. The rule used to score each action (with the steering angle and driving throttle combined) is shown in Extended Data Fig. 5b. The score of the neural network is the summation of the scores over the entire test dataset of the network normalized by that of the

traditional approach.

Image classification task of learning a new class

We demonstrate the improvement learning of a new class in the MNIST image classification task, which involves images of the digits 0 and 2-9 for off-chip training and images of the digit 1 for on-chip learning. A multilayer neural network with dimensions of 784 × 100 × 9 (Fig. 4f) is constructed during off-chip training, and this network is then transferred to the memristor chip during the weight transfer stage. The neural network is expanded to dimensions of 784 × 100 × 10 with a newly added output neuron for the new class. The expanded parts of the conductance weights are programmed to the HRS at the differential rows for the new class by using single-side verification. Only the expanded conductance weights are updated, while the old conductance weights are frozen during on-chip learning. The freezing of the old conductance weights is realized by using the obtained inference results as targets for the nine old output neurons. The on-chip learning is performed on 150 images randomly selected from the training set of the new class (6,742 images). The inference is performed on the test sets of the old classes and the new class at intervals of 10 training images; the inference is conducted on 1,000 randomly selected images acquired from the test set of the old classes and 1,135 images (the entire test set) of the new class.

Audio recognition task of learning new samples

We demonstrate the learning of new samples in a speech recognition task conducted on the AudioMNIST dataset[44], which consists of male and female audio samples of spoken digits

(0-9). The dataset consists of 30,000 audio samples of spoken digits (0-9) provided by 48 men and 12 women, with each speaker saying each digit 50 times. Mel-frequency cepstral coefficient (MFCCs) are used as the features for the audio samples, while a two-layer perceptron with dimensions of 392 × 100 × 10 is used to classify the MFCC features. As illustrated in Extended Data Fig. 6a, the improvement learning of female audio samples is based on the model trained with male audio samples. First, the base model is off-chip trained with the 6,000 samples of the first 12 men and then deployed on the memristor chip. Next, the improvement learning of new samples is performed on the chip with 5,400 samples (the first 45 audio samples of each woman).

Extended Data Fig. 6b shows the evolution trends of the model accuracies for male and female audio during improvement learning. The on-chip inference is performed on the 18,000 samples of the last 36 men and 600 samples of the last 5 audio samples of each woman. After improvement learning with the female audio samples, the accuracy of the model for female audio increases from 85.7% to 89.5%, while the accuracy of the model for male audio slightly decreases from 91.4% to 91%.

Motion control task of learning a new class

We also demonstrate the learning of a new class in another motion control task involving the abovementioned light-chasing car platform. In this task, improvement learning aims to make the car learn to move backwards, and both the steering angle and driving throttle are taken as the classification results of the neural network. As illustrated in Extended Data Fig. 7a, the improvement learning of the unreachable situation is based on the model developed for

the reachable situation. The unreachable situation refers to the data whose camera images have a light spot at their two bottom corners. In this case, the car will not reach the target (i.e., light spot) even though it turns the steering motor to the rightmost (or leftmost) setting and moves forward until it stops. After improvement learning of the unreachable situation, whose driving throttle is labelled “move backwards”, the car learns to move backwards to align itself to the target in this case. The behaviours of the car before and after improvement learning are illustrated in Extended Data Fig. 7b as the trajectories of the weight transfer and improvement learning, respectively, and are shown in Supplementary Video 2. A sequence of camera images corresponding to the trajectory after improvement learning is shown in Extended Data Fig. 7d. The driving throttle classification accuracies achieved for the old classes and the new class over the course of the improvement learning are shown in Extended Data Fig. 7c.

Data availability

The MNIST dataset[30] and AudioMNIST dataset[44] are publicly available. The dataset for the light-chasing car tasks is available from the corresponding author upon reasonable request. The light-chasing car used in this work is developed based on the open-source donkeycar platform. The data that support the findings of this study are available from the corresponding author upon reasonable request.

Code availability

The codes that support the findings of this study are available from the corresponding author upon reasonable request.

Acknowledgements

This work is supported in part by the Minstry of Science and Technology (MOST) of China (2021ZD0201200), the National Natural Science Foundation of China (92064001, 62025111), the XPLORER Prize, the IoT Intelligent Microsystem Center of Tsinghua University-China Mobile Joint Research Institute, and the Beijing Advanced Innovation Center for Integrated Circuits.

Author Contributions

B.G., Y.W., and H.W. supervised this project and proposed the overall architecture. W.Z., B.G., and P.Y. contributed to the whole experiment design. P.Y., Q.L., D.W., and D.W. designed the circuits. B.G., J.T., H.Q., and H.W. contributed to the memristor device fabrication and integration with CMOS. W.Z., P.Y., and Q.L. contributed to the implementation of the test system. Q.Z. worked on the development of AI models. W.Z. implemented the AI models on the chip and conducted the chip and system measurements. W.Z., X.M., and Y.Z. performed the simulation and benchmark. W.Z., B.G., and Y.P. contributed to the writing and editing of the manuscript. All authors discussed the results and reviewed the manuscript.

Competing Interests

The authors declare no competing interests.

Materials & Correspondence

Correspondence to Bin Gao (gaob1@tsinghua.edu.cn) and Huaqiang Wu (wuhq@tsinghua.edu.cn).

References

1. Shi, W., Cao, J., Member, S., Zhang, Q. & Member, S. Edge Computing : Vision and Challenges. **3**, 637–646 (2016).
2. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).
3. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
4. Coates, A. *et al.* Deep learning with COTS HPC systems. *30th Int. Conf. Mach. Learn. ICML 2013* 2374–2382 (2013).
5. Horowitz, M. 1.1 Computing's energy problem (and what we can do about it). in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)* vol. 57 10–14 (IEEE, 2014).
6. Wong, H. S. P. & Salahuddin, S. Memory leads the way to better computing. *Nat. Nanotechnol.* **10**, 191–194 (2015).
7. Strubell, E., Ganesh, A. & McCallum, A. Energy and policy considerations for modern deep learning research. *AAAI 2020 - 34th AAAI Conf. Artif. Intell.* 1393–13696 (2020).
8. Ielmini, D. & Wong, H. S. P. In-memory computing with resistive switching devices. *Nat. Electron.* **1**, 333–343 (2018).
9. Zidan, M. A., Strachan, J. P. & Lu, W. D. The future of electronics based on memristive systems. *Nat. Electron.* **1**, 22–29 (2018).

10. Liu, Q. *et al.* 33.2 A Fully Integrated Analog ReRAM Based 78.4TOPS/W Compute-In-Memory Chip with Fully Parallel MAC Computing. in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)* 500–502 (IEEE, 2020).
11. Wan, W. *et al.* 33.1 A 74 TMACS/W CMOS-RRAM Neurosynaptic Core with Dynamically Reconfigurable Dataflow and In-situ Transposable Weights for Probabilistic Graphical Models. in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)* 498–500 (IEEE, 2020).
12. Xue, C. *et al.* A CMOS-integrated compute-in-memory macro based on resistive random-access memory for AI edge devices. *Nat. Electron.* **4**, 81–90 (2021).
13. Chen, W. H. *et al.* CMOS-integrated memristive non-volatile computing-in-memory for AI edge processors. *Nat. Electron.* **2**, 420–428 (2019).
14. Ambrogio, S. *et al.* Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature* **558**, 60–67 (2018).
15. Sebastian, A., Le Gallo, M., Khaddam-Aljameh, R. & Eleftheriou, E. Memory devices and applications for in-memory computing. *Nat. Nanotechnol.* **15**, 529–544 (2020).
16. Zhang, W. *et al.* Neuro-inspired computing chips. *Nat. Electron.* **3**, 371–382 (2020).
17. Prezioso, M. *et al.* Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* **521**, 61–64 (2015).
18. Yao, P. *et al.* Face classification using electronic synapses. *Nat. Commun.* **8**, 1–8 (2017).
19. Li, C. *et al.* Efficient and self-adaptive in-situ learning in multilayer memristor neural networks. *Nat. Commun.* **9**, 7–14 (2018).

20. Wang, Z. *et al.* Reinforcement learning with analogue memristor arrays. *Nat. Electron.* **2**, 115–124 (2019).

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

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

23. Cai, F. *et al.* A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations. *Nat. Electron.* **2**, 290–299 (2019).

24. Burr, G. W. *et al.* 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).

25. Wu, H. *et al.* Device and circuit optimization of RRAM for neuromorphic computing. in *2017 IEEE International Electron Devices Meeting (IEDM)* 11.5.1-11.5.4 (IEEE, 2017).

26. Chen, P. Y., Peng, X. & Yu, S. NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning. *IEEE Trans. Comput. Des. Integr. Circuits Syst.* **37**, 3067–3080 (2018).

27. Xi, Y. *et al.* In-memory Learning with Analog Resistive Switching Memory: A Review and Perspective. *Proc. IEEE* **109**, 14–42 (2021).

28. Zhang, Q. *et al.* Sign backpropagation: An on-chip learning algorithm for analog RRAM neuromorphic computing systems. *Neural Networks* **108**, 217–223 (2018).

29. Wu, W. *et al.* A Methodology to Improve Linearity of Analog RRAM for

Neuromorphic Computing. in *2018 IEEE Symposium on VLSI Technology* vols 2018-June 103–104 (IEEE, 2018).

30. Bengio, Y. & Haffner, P. Gradient-Based Learning Applied to Document Recognition. **86**, (1998).

31. Chi, P. *et al.* PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory. *Proc. - 2016 43rd Int. Symp. Comput. Archit. ISCA 2016* 27–39 (2016).

32. Song, L., Qian, X., Li, H. & Chen, Y. PipeLayer: A Pipelined ReRAM-Based Accelerator for Deep Learning. in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)* 541–552 (IEEE, 2017).

33. Khaddam-Aljameh, R. *et al.* HERMES Core – A 14nm CMOS and PCM-based In-Memory Compute Core using an array of 300ps/LSB Linearized CCO-based ADCs and local digital processing. in *2021 Symposium on VLSI Circuits* 1–2 (IEEE, 2021).

34. Wan, W. *et al.* A Voltage-Mode Sensing Scheme with Differential-Row Weight Mapping for Energy-Efficient RRAM-Based In-Memory Computing. in *2020 IEEE Symposium on VLSI Technology* vols 2020-June 1–2 (IEEE, 2020).

35. Hung, J.-M. *et al.* A four-megabit compute-in-memory macro with eight-bit precision based on CMOS and resistive random-access memory for AI edge devices. *Nat. Electron.* **4**, 921–930 (2021).

36. Jouppi, N. P. *et al.* In-Datacenter Performance Analysis of a Tensor Processing Unit. in *Proceedings of the 44th Annual International Symposium on Computer Architecture* vol. Part F1286 1–12 (ACM, New York, NY, USA, 2017).

37. Oh, T., Maghari, N. & Moon, U. K. A 5MHz BW 70.7dB SNDR noise-shaped two-step quantizer based ΔΣ ADC. *IEEE Symp. VLSI Circuits, Dig. Tech. Pap.* 162–163 (2012).

38. Chen, J. *et al.* A parallel multibit programing scheme with high precision for RRAM-based neuromorphic systems. *IEEE Trans. Electron Devices* **67**, 2213–2217 (2020).

39. Li, F., Zhang, B. & Liu, B. Ternary Weight Networks. (2016).

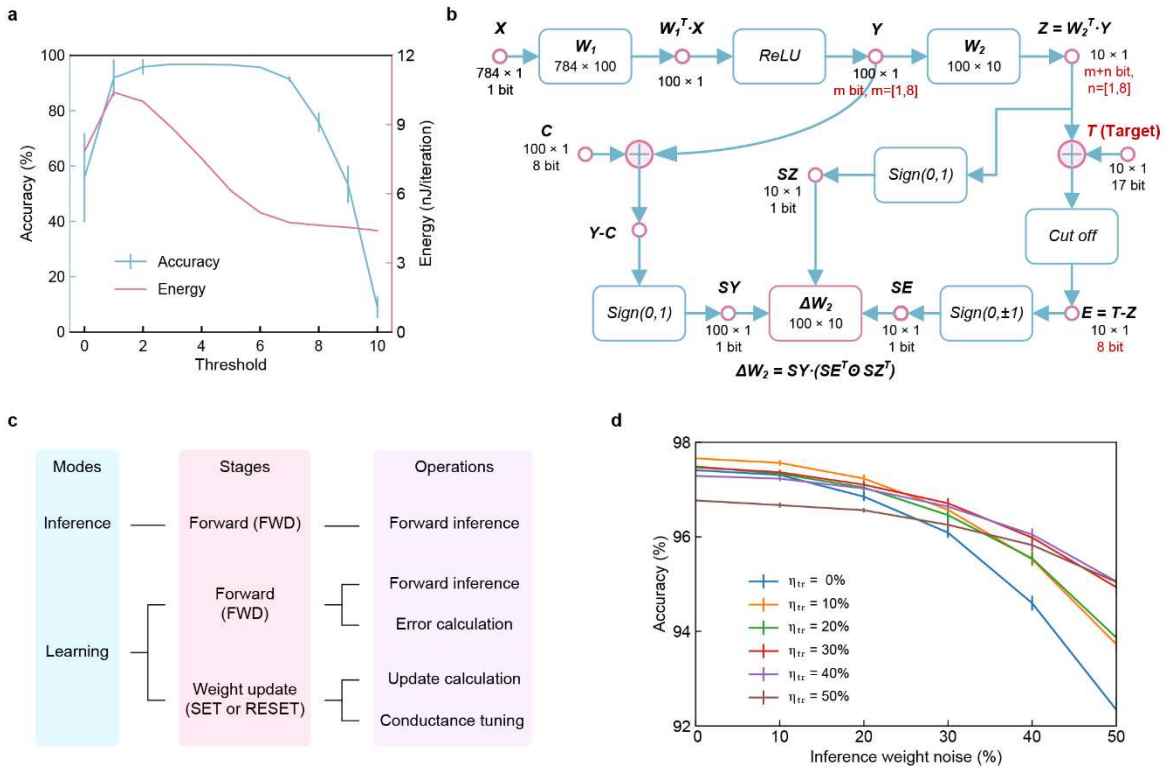
40. Joshi, V. *et al.* Accurate deep neural network inference using computational phase-change memory. *Nat. Commun.* **11**, 1–13 (2020).

41. O'Connor, M. Highlights of the High- Bandwidth Memory (HBM) Standard. *Mem. Forum* (2014).

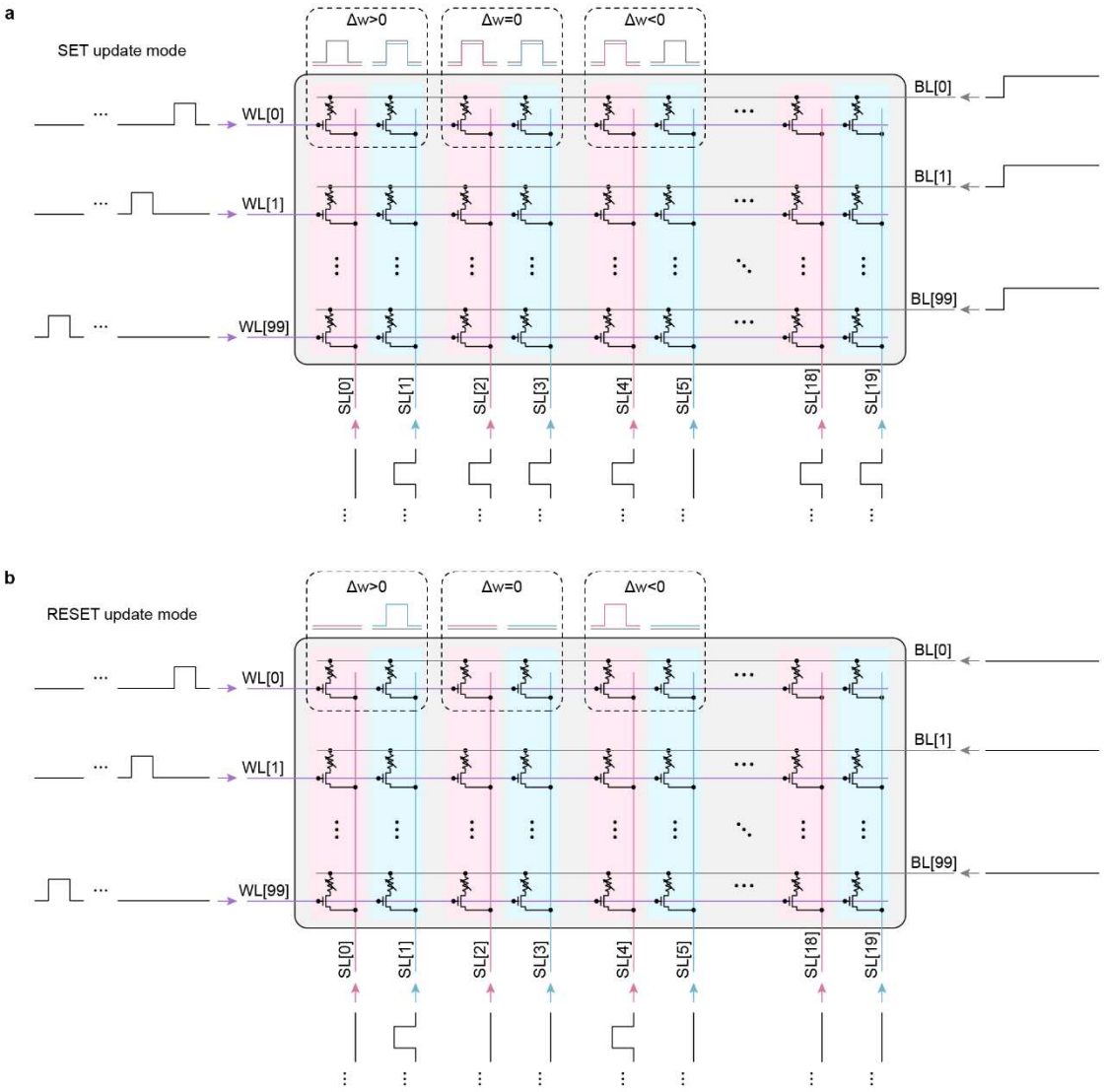
42. Arunkumar, A. *et al.* MCM-GPU: Multi-Chip-Module GPUs for Continued Performance Scalability. in *Proceedings of the 44th Annual International Symposium on Computer Architecture* vol. 45 320–332 (ACM, New York, NY, USA, 2017).

43. Bojarski, M. *et al.* End to End Learning for Self-Driving Cars. 1–9 (2016).

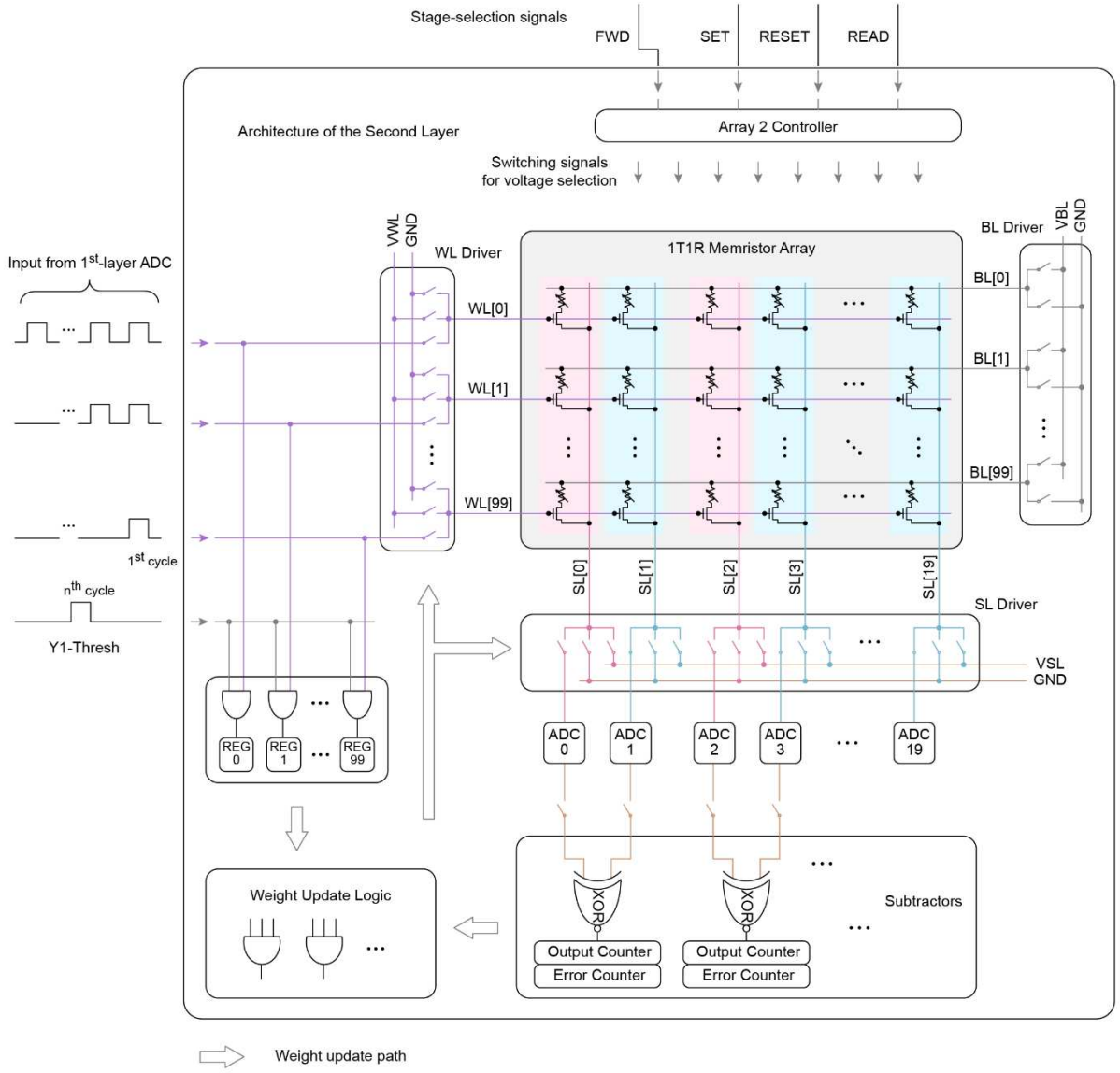
44. Becker, S., Ackermann, M., Lapuschkin, S., Müller, K.-R. & Samek, W. Interpreting and Explaining Deep Neural Networks for Classification of Audio Signals. 2–6 (2018).



Extended Data Fig. 1 | Details of the algorithm for the STELLAR architecture and the results of noise-aware training. a, The role of the threshold in the STELLAR algorithm, showing the effects of different thresholds on classification accuracy and energy consumption of on-chip learning. **b,** Schematic of the modified algorithm for chip implementation. *m* and *n*, ranging from 1 to 8, represent the resolutions of the ADCs of the 1st and 2nd layers, respectively. **c,** Dendrogram of the chip's working modes and the corresponding stages and operations. The forward stage involves a forward inference operation and an error calculation operation. The weight update stage involves a weight update calculation and a conductance tuning operation. **d,** Classification accuracies as a function of the inference weight noise for the off-chip-trained models with different levels of weight noise injected during training.



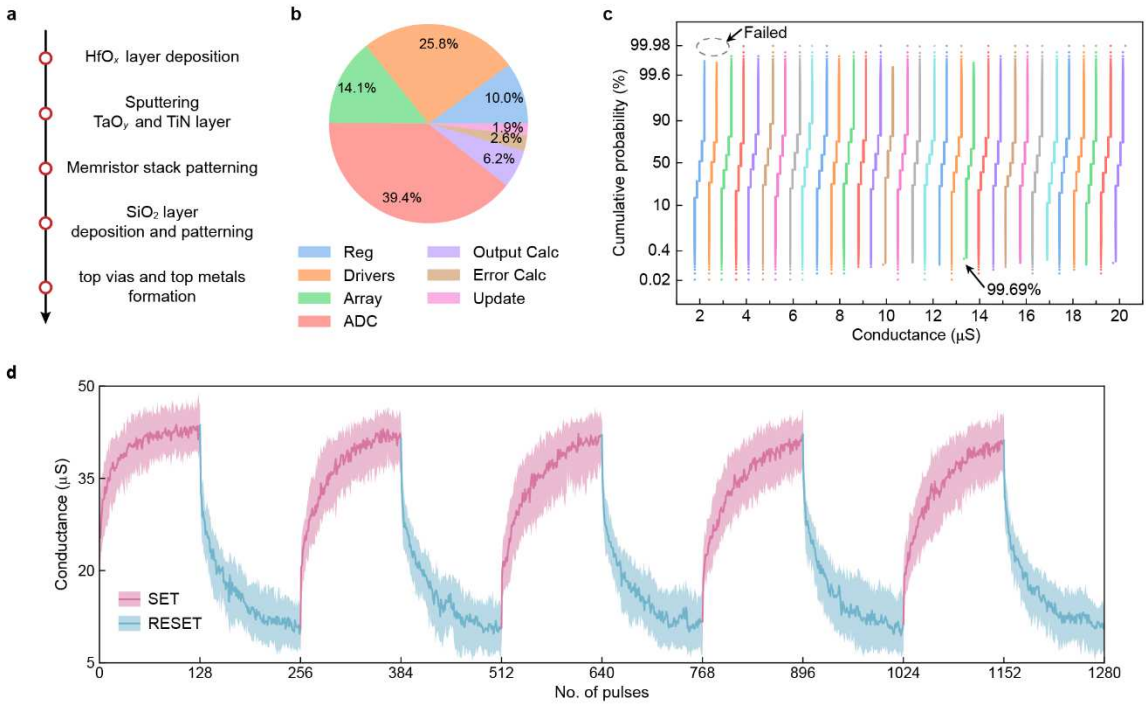
Extended Data Fig. 2 | Schematics of array operations during update modes. Schematics of array operations in SET update modes (**a**) and RESET update modes (**b**). The WL signals select which row to operate in turns, while the SL signals determine which memristors to operate on the selected row.



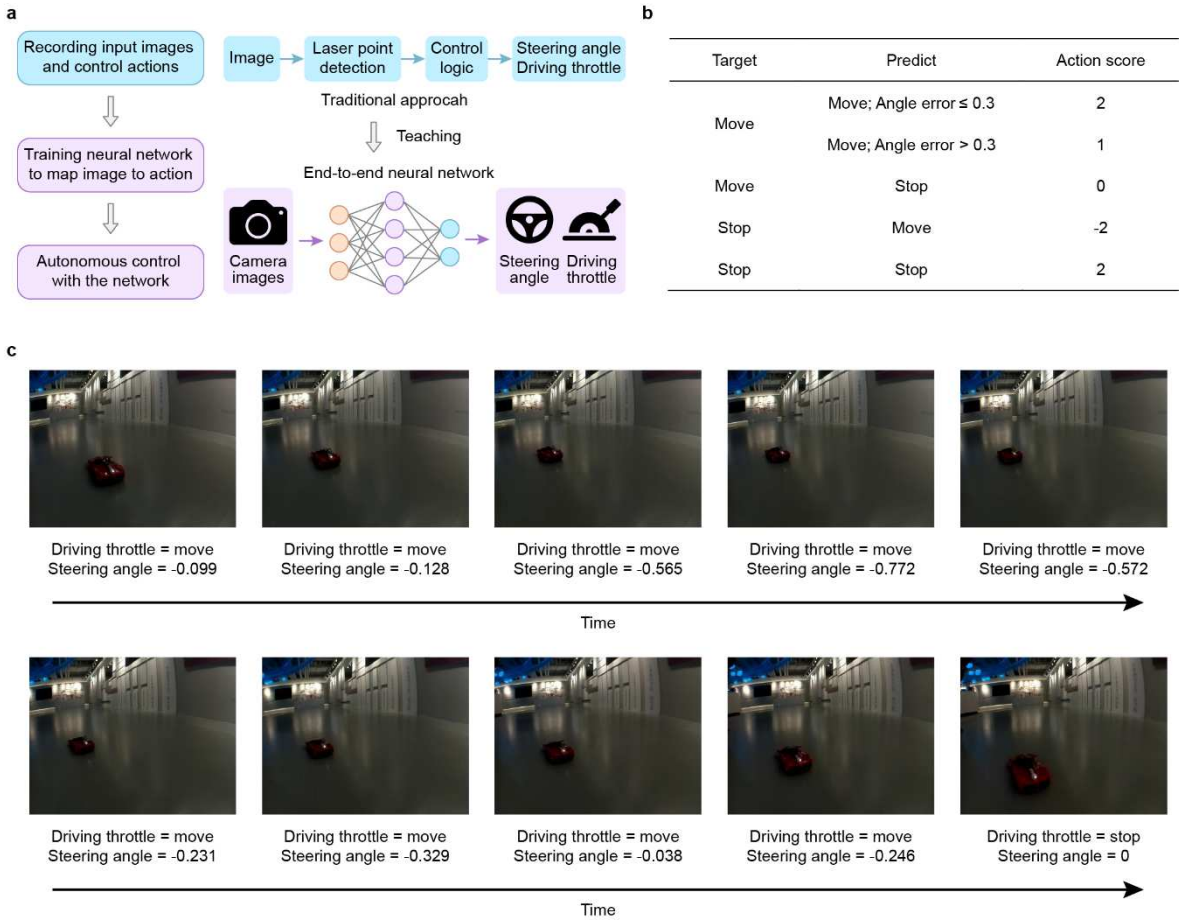
794

795 **Extended Data Fig. 3 | Architecture of the 2nd layer of the memristor chip.**

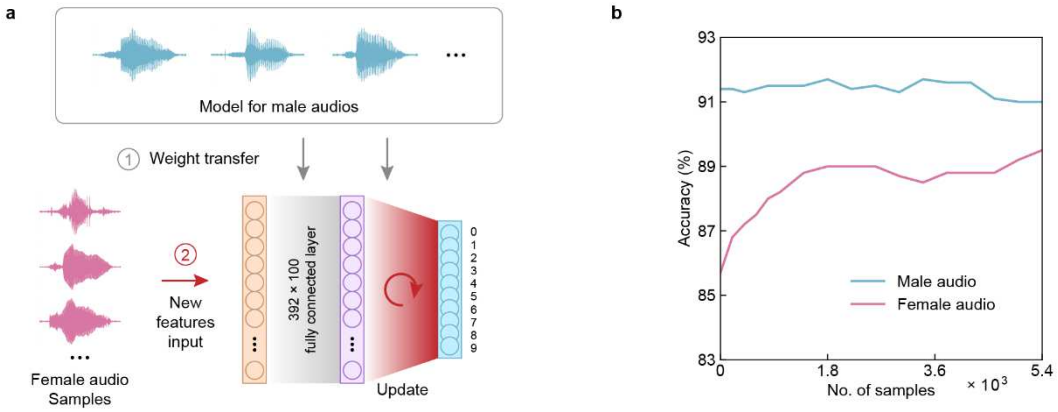
796



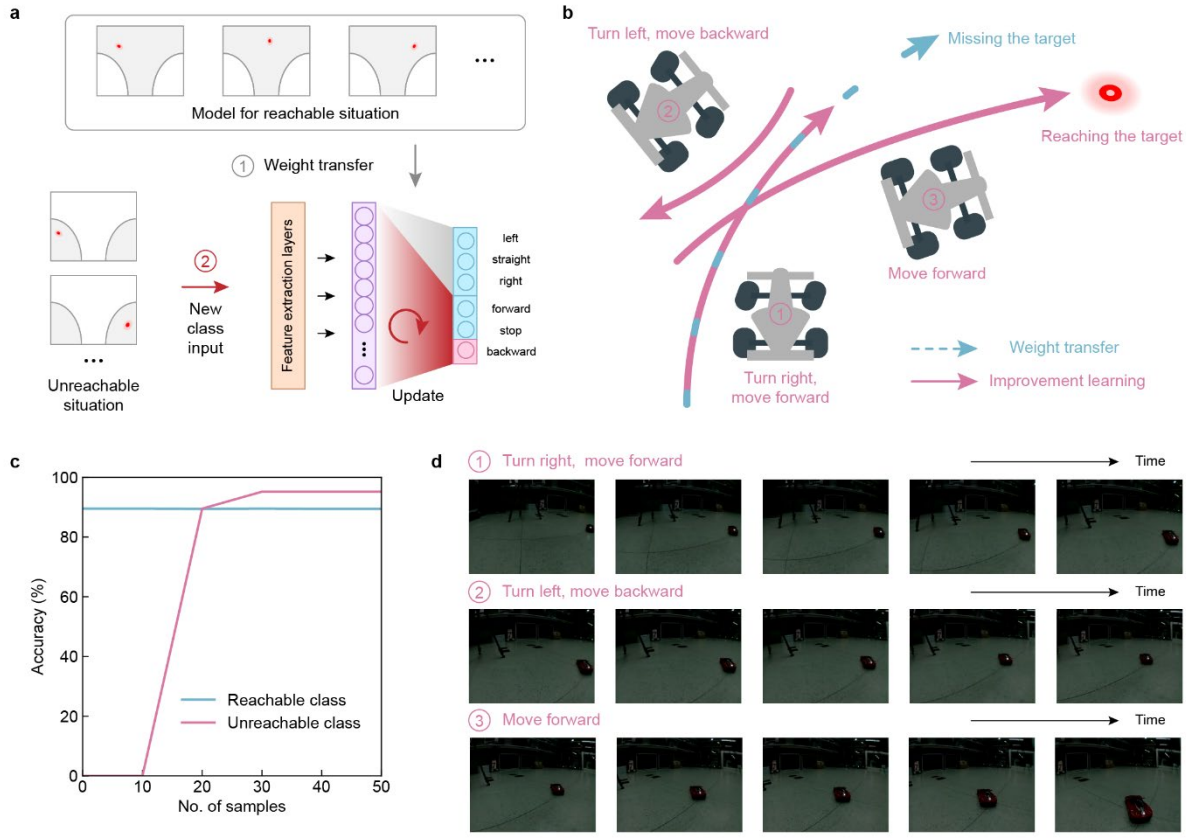
Extended Data Fig. 4 | Fabrication and characterization of the memristor devices. a, Process flow of the fabrication of the memristor devices. **b**, Chip area breakdown. **c**, Cumulative probability distribution of ~160K memristors after programming them to 32 conductance states. The memristors outside of the target range are not shown, as marked by the grey circle. The minimum success rate among the 32 conductance states is 99.69%. **d**, Analogue switching behaviours of 64 memristors under identical pulse trains. The dark lines represent the median values of the conductance, while the light colours fill the regions between the lower quartiles and the upper quartiles.



Extended Data Fig. 5 | Motion control task of learning new samples. a, Diagram of the realization of end-to-end autonomous control. **b**, Table of the rule scores for different control decisions under the consideration of both the steering angle and the driving throttle. Taking the first row as an example, for a certain network prediction, if the predicted throttle and the corresponding labelled throttle are both "move" and the difference between the predicted angle and the labelled angle is no larger than 0.3, the score of this predicted control decision is 2. **c**, A sequence of input camera images and their corresponding control decisions produced during real-time driving by the network deployed on the memristor chip.



Extended Data Fig. 6 | Audio recognition task of learning new samples. a, Illustration of the improvement learning of new samples in the audio recognition task. **b**, Changes in the recognition accuracies achieved for male audio and female audio over the course of the on-chip improvement learning female audio samples.



820

821 **Extended Data Fig. 7 | Motion control task of learning a new class. a**, Illustration of the improvement
822 learning of new class in the light-chasing car task. **b**, Illustration of the car's movements and trajectories
823 before and after improvement learning of the new class. Before improvement learning, the car moves forward
824 even though it will miss its target. After improvement learning, the car moves backwards to align itself with
825 the target and then moves forward to reach the target. **c**, Changes in the throttle classification accuracies for
826 the old classes and the new class over the course of the on-chip improvement learning the new class. **d**. A
827 sequence of camera images corresponding to the trajectory produced after improvement learning and shown
828 in (**b**). The car first moves forward until it will miss the target, then moves backwards to align itself with the
829 target, and finally moves forward again to reach the target.

830

Extended Data Table 1 | Detailed metrics produced during each stage of the learning mode.

	Delay/μs	Power/mW	Energy/nJ
FWD stage	14.85	54.64	811.3
SET stage	85.95	2.49	213.7
RESET stage	55.95	3.01	168.1

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- SupplementaryVideo1.mp4
- SupplementaryVideo2.mp4
- SupplementaryVideo3.mp4