ADVANCED INTELLIGENT SYSTEMS
Open Access
www.advintellsyst.com

Energy-Efficient Memristive Euclidean Distance Engine for Brain-Inspired Competitive Learning

Houji Zhou, Jia Chen, Yinan Wang, Sen Liu, Yi Li, Qingjiang Li,* Qi Liu, Zhongrui Wang, Yuhui He, Hui Xu, and Xiangshui Miao*

Inspired by competitive rules of the nature, competitive learning contributes to the specialization of the human brain and the general creativity of mankind. However, the construction of hardware competitive learning neural network still faces great challenges due to the lack of an accurate distance computation method and a self-adaptive in situ training scheme. Herein, a fully memristive Euclidean distance (ED) engine based on analog multiply-accumulate operation in a 32 × 32 TiN/TaO_x/HfO_x/TiN 1T1R array is demonstrated. The dual-layer devices perform multilevel modulation under the target-aware programming method with excellent read linearity in a dynamic range of 10–100 μS. The ED calculation is verified experimentally on a test board with an *O*(1) temporal complexity. Furthermore, in situ training and offline inference schemes for competitive learning, based on the ED engine, are developed and the simulated results show comparable success rates with those obtained by the CPU-based software. Compared with a state-of-the-art RTX6000 GPU (0.5 TOPS W^{-1}), the energy efficiency of competitive learning models on ED engines can yield 100× improvements by utilizing optimized memristive devices.

1. Introduction

Competitive rules are widely observed in the nature, which dictates the evolution of organisms and cells via natural selection.[1] In the brain, a competition mechanism exists among neurons wherein synaptic connections with high spiking frequencies and strong inputs are retained and strengthened, while the connections with low frequencies and weak inputs are pruned or decayed[2–4] (**Figure 1**a). Inspired by the information processing mechanisms of the biological brain, competitive learning neural networks (CLNNs) have received widespread attention.[5–9] Their corresponding network structure is shown in Figure 1b.[10] As a traditional artificial neural network (ANN) model, CLNN is used to discover patterns in the distribution of data mainly through unsupervised learning, based on similarity measurements between input samples and weight vectors.[11–14] In the era of artificial intelligence and the Internet-of-Things (AIoT), similarity measurements are commonly accepted in machine learning algorithms, and are extensively used in recommendation systems, pattern recognition, data queries, and other applications.[15–19] Euclidean distance (ED) is one of the most well-established methods for similarity measurements and has been frequently applied for image recognition, natural language processing, data mining, wireless localization, and other applications.[20–24] It is used to quantitatively represent the distance between two vectors in Euclidean space, as shown in Figure 1c. The absolute value of the ED calculation can be used to determine the degree of similarity, and the sample with the smallest ED value is considered to yield the best match.

With the dramatic increase in data dimensions in the AIoT era, ED-based applications have encountered huge challenges on the resource-constrained edge computing platforms due to serious bottlenecks in computing power and efficiency.[25] Memristive in-memory computing has emerged as a promising solution for energy-efficient non-von Neumann computing paradigms.[26–28] The frequent vector-matrix multiplication (VMM) operations in ANNs, such as multilayer perceptrons and convolutional neural networks, have been accelerated considerably owing to in-memory computing in which multiply-accumulate (MAC) operations can be executed in a single step using Ohm's and Kirchhoff's laws in a memristive crossbar array.[29–34]

H. Zhou, J. Chen, Y. Li, Y. He, X. Miao
Wuhan National Laboratory for Optoelectronics & School of Optical and Electronic Information
Huazhong University of Science and Technology
Wuhan, China
E-mail: liyi@hust.edu.cn

Y. Wang, S. Liu, Q. Li, H. Xu
College of Electronic Science and Technology
National University of Defense Technology
Changsha, China
E-mail: qingjiangli@nudt.edu.cn

Q. Liu
School of Microelectronics
Fudan University
Shanghai, China

Z. Wang
Department of Electrical and Electronic Engineering
The University of Hong Kong
Hong Kong

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/aisy.202100114.

© 2021 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH. This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

DOI: 10.1002/aisy.202100114

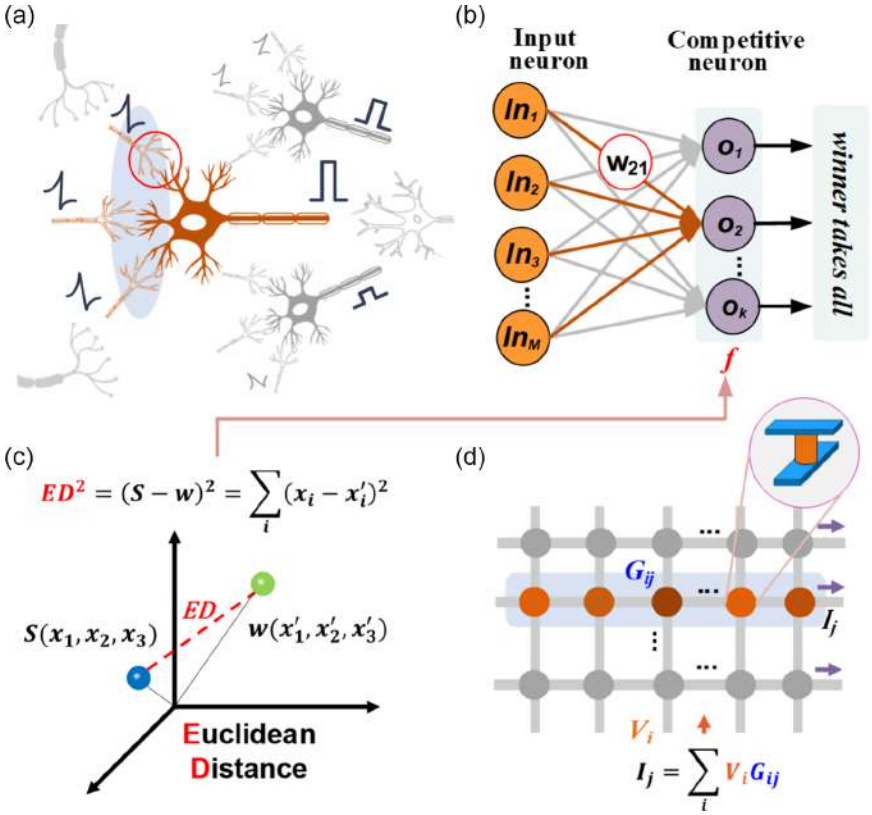

Figure 1. Schematic concepts of competitive learning. a) Illustration of neuronal competition in the brain. Input signals stimulate the receptor neurons. The neuron with the heaviest weight connection outputs a spike with a higher intensity and suppresses the other channels. b) Structure of competitive learning model with WTA as learning rule in which the distance (similarity) calculation plays the key role for the competitive process. c) Illustration of ED in Euclidean space, which is essential for competitive learning. A point in an N-dimensional Euclidean space is represented by an N-dimensional vector. d) Single-step vector-matrix multiplication operation on the memristor array owing to Ohm's and Kirchhoff's laws.

Therefore, for CLNNs, building an ED engine utilizing a single-clock VMM operation in memristor arrays (Figure 1d) is essential to overcome traditional computational limitations. Currently, building a functional, fully memristive ED engine is challenging, although there are initiatives to implement ED calculations on memristor arrays.[35–38] Specifically, the remaining issues are mainly the lack of 1) complete expression for ED calculation on a hardware platform, 2) efficient in situ training scheme for CLNNs with hardware ED engines, and 3) remarkable versatility to different ED-based algorithms.

In this study, a fully memristive ED engine was demonstrated for the first time that exhibited large hardware computational efficiency and flexibility. ED calculations for data with five dimensions were implemented on a TiN/TaO_x/HfO_x/TiN 1T1R array to verify the reliability of the ED engine. The favorable analog behavior and the excellent dynamic-range read linearity of the memristor cells ensure the accurate data mapping as well as precise analog computing results. By utilizing the memristive ED engine, CLNNs were demonstrated in prototype clustering tasks. With in situ training and optional offline inference schemes, the clustering task based on the memristive ED engine yielded equivalent results for the IRIS and the breast cancer datasets compared with that running on full-precision software. The memristive ED engine provides a vigorous and general solution for the hardware implementation of competitive learning, which completes ED calculations within constant time and features in fully hardware online weight updating.

2. Results and Discussion

2.1. Principle of Memristive Euclidean Distance Engine

Mathematically, the ED of two data vectors S and w is calculated using **Equation (1)**

$$\begin{cases} D(S, w) = |S - w|^2 \\ \quad = S^2 - 2S \cdot w + w^2 \\ S = [s_1, s_2, \ldots, s_m] \\ w = [w_1, w_2, \ldots, w_m] \end{cases} \quad (1)$$

where S and w are the m-dimensional vectors in Euclidean space. In CLNNs, S represents the set of sample vectors to be classified, and w represents the weight vector of the networks. From Equation (1), the ED calculation contains the dot product term ($-2S \cdot w$) and two non-negligible squared terms (S^2, w^2). Some specific competitive learning algorithms, such as self-organizing maps [35] and K-means,[36] have been implemented on the memristor crossbar array by ignoring one or two squared terms.

© 2021 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH

26404567, 2021, 11, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/aisy.202100114 by CochraneItalia, Wiley Online Library on [30/04/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Sheridan et al.[37] revealed that the ED calculation was simplified in the form of a dot product of vectors for pattern-matching tasks. The intensive MAC operations were then accelerated by memristor arrays by mapping weights to memristor conductance. However, in case the vectors are not normalized, the dot product term could not accurately express the true distance between vectors anymore. Subsequently, Jeong et al.[36] experimentally proposed a scheme for the direct comparison of the Euclidean distances without normalizing the weights on a memristor crossbar. The squared term of each input sample vector is ignored in this scheme because it is a constant for all weight vectors. Thus, a direct, single-step comparison of the ED can be implemented on a memristor array by adding an extra row on the array to store the squared term of the weight vectors. Notably, this improved ED calculation solution can only perform the forward inference of CLNN tasks, whereas the process of online updating, which leads to the fully hardware implementation and self-adaption of the network, is almost hardly achieved on the memristor. In addition, this scheme is only suitable for applications that require a comparison of the relative ED values while not for absolute ED values. Therefore, it is of great importance to compute the full expression of ED in memristor arrays and to implement online weight updating. Herein, we focus on these problems by devising a fully memristive ED engine. Based on the traditional CPU architecture, the basic operation for ED calculation involves a serial subtractor, multiplier, adder, and accumulator, which can be replaced by the memristive ED engine in a single-clock step (**Figure 2**a). First, Equation (1) can be rewritten as Equation (2)

$$\begin{aligned} D(S,w) &= S^2 - 2S\cdot w + w^2 \\ &= S\times(S-w) + (-w)\times(S-w) \end{aligned} \tag{2}$$

By taking $(S - w)$ as a whole, the ED calculation can be converted into a sum of two dot product terms. Accordingly, based on the principle of **Equation (2)**, a fully memristive ED engine was designed, as shown in Figure 2b. The item $(S - w)$ was mapped to the differential conductance rows in the memristor array. The elements of the two vectors were mapped as the memristive conductance in the two rows. The dimension of the vectors determined the number of columns of the array. The subtraction of the two vectors was then achieved by the differential conductance pairs, which is the a common method used to implement negative weights in various memristive neural networks.[39–41] To achieve the summation of the two dot product terms in Equation (2), the two horizontal $(S - w)$ conductance kernels are iteratively programmed, and other vectors, namely, S and $-w$, were mapped as input voltage signals (encoded as the number of voltage pulses with a fixed amplitude, or single pulses of varying amplitudes to denote different vector elements) and multiplied by each of the two conductance kernels. Therefore, similar to distributed kernels,[34] absolute ED calculations can be completely expressed on memristor arrays to leverage the parallelism of the VMM operations to achieve a single-step operation. The time complexity of the ED calculation is lowered from $O(n)$ for CPU calculations to $O(1)$ for our designed memristive ED engine. That is, the feedforward process of CLNNs that utilize the ED calculation can be accelerated directly on the memristor arrays. Moreover, the fully calculated EDs can find a wide spectrum of applications, such as kernel functional calculations and wireless locations.[42–44]

Furthermore, the designed ED engine was also fitted to the online updating rule of competitive learning tasks. In CLNNs based on the learning rule of Winner Takes All (WTA), only one winning neuron updates its weight while the other neurons

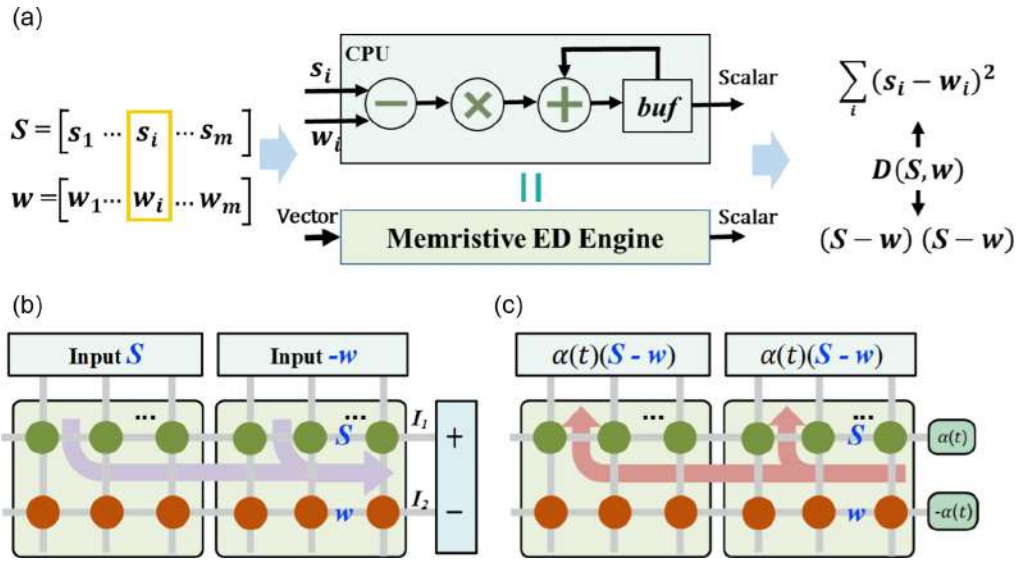

Figure 2. Illustration of the designed ED engine in the memristor array. a) Computing complexity of the memristive ED engine can be reduced to $O(1)$, compared with $O(m)$ in traditional CPU. b) Forward step associated with the calculation of ED on the memristive ED engine. The two stored vectors, S and w, are programmed twice in the horizontal direction on the memristor array in a repeated manner. The difference of the output currents indicates the ED result between S and w. c) Backward read process used to obtain the updated values for competitive learning. By applying voltage signals proportional to $\alpha(t)$ and $-\alpha(t)$ to the corresponding row, the output currents suggest the desired updated values determined by the gradient descent method.

© 2021 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH

26404567, 2021, 11, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/aisy.202100114 by Cochraneltalia, Wiley Online Library on [30/04/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

maintain their values. The updating rule can be demonstrated by Hebb's rule and expressed according to Equation (3)[45,46]

$$\Delta w = \alpha(t)(v_{In} - v_w) \quad (3)$$

where $\alpha(t)$ denotes the learning factor of the network, v_{In} and v_w, respectively, indicate the input and the weight vectors that correspond to the input pattern and winning neuron used for the update. In general, the vectors in the sample set (S) are used as the input pattern, and the weight vectors (w) are used as the weight map for CLNNs. Therefore, as shown in Figure 2c, a backward online update can also be performed on the designed memristive ED engine. The learning factor $\alpha(t)$ was encoded with a fixed voltage amplitude, which was input to the rows storing S and w with positive and negative pulses, respectively. This reverse VMM operation mapped exactly the calculation of Equation (3). Therefore, in the designed memristive ED engine, the complete expression of ED calculation was associated with high computational parallelism and low time complexity, and an online update for CLNNs also became possible.

2.2. Experimental Demonstration of a Memristive ED Engine

To experimentally verify the feasibility of the designed memristive ED engine, an field programmable gate array (FPGA) test board consisting of a 32×32 1T1R TiN/TaO_x/HfO_x/TiN memristor array was used and the packaged 1T1R array is shown **Figure 3**a. The optical image of the test board is shown in Figure S1, Supporting Information. The 1T1R cell was formed

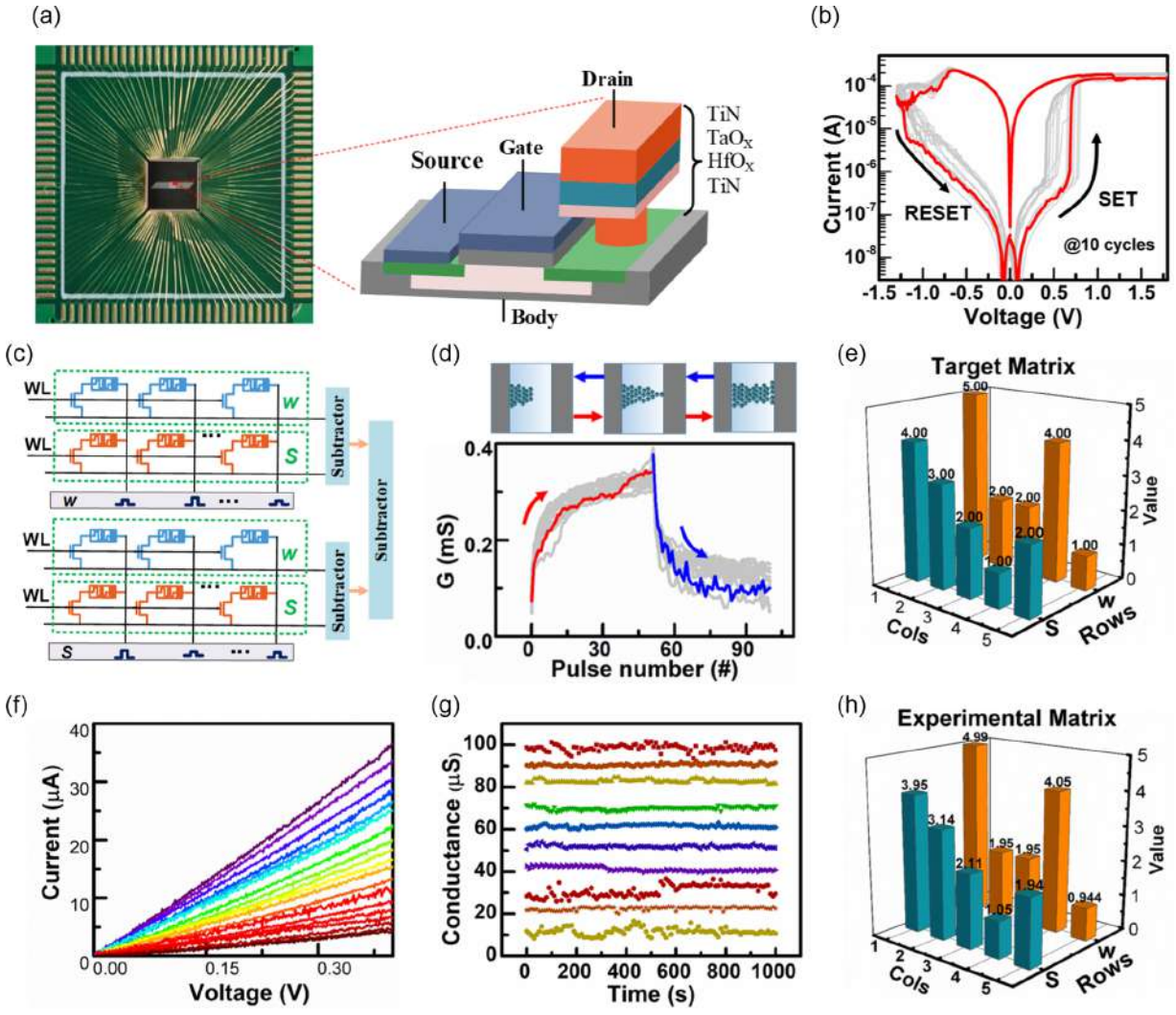

Figure 3. Experimental implementation of ED engine on a 1T1R memristor crossbar array. a) The packaged memristor array with the TiN/HfO_x/TaO_x/TiN 1T1R cell. b) Typical resistive switching characteristics of a 1T1R memristor cell. c) The testing structure of the proposed ED engine in a 1T1R array. d) Pulse-induced conductance tuning behavior exhibits its potential for use in analog computing. e) The target matrix to be programmed in the array. The conductance will be programmed by the target-aware method to achieve the target values. f) The linear I–V relationship for a read voltage of 0.4 V. It proves that the encoded input voltage below 0.4 V does not affect the conductance state. g) Ten-level retention properties for the measured memristor. h) Experimental programmed values on the 1T1R test board. The maximum error for programming is less than 6% while the average error is approximately 2.8%.

© 2021 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH

26404567, 2021, 11, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/aisy.202100114 by CochraneItalia, Wiley Online Library on [30/04/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

by growing a TiN/TaO_x/HfO_x/TiN memristor on the drain side of an N-type metal–oxide–semiconductor transistor, and the details of fabrication are illustrated in the Experimental Section. Typical bipolar resistive switching characteristics of a 1T1R memristor cell are shown in Figure 3b. Utilizing the transistor as a selector and a current limiter, 1T1R arrays are immune to the sneak path issue. The SET operation applied positive voltages to the gate and drain, while RESET applied positive voltages to the gate and source. Notably, negative drain or source voltages are usually not available for 1T1R operations owing to the limit of transistors. Subject to the constraint, the implementation of the designed memristive ED engine on a 1T1R array is shown in Figure 3c. Differential pairs of S and w were stored in two separate kernels, and the positive voltage signals with different amplitudes, encoded from S and w, and then inputted to each of the two kernels. Finally, the two differential currents are passed through a subtractor, and the output current was proportional to the actual calculated ED value of the array. Our 1T1R cells have demonstrated a continuously tuneable conductance, as shown in Figure 3d, which indicates the analogue computing capability. The conductive filament of the memristor cells will be enhanced or weakened under SET or RESET pulses, respectively, and thus resulting in cycling potentiation or depression behaviors. Taking the target matrix shown in Figure 3e as an example, the ED calculation was proved on the 1T1R array. The mapping lists for memristive conductance and input voltage amplitudes are shown in Table SI, Supporting Information. The voltage amplitudes from 0 to 0.4 V were encoded vectors, and the linear I–V relationship has been observed on memristors of different conductance (Figure 3f), yielding accurate readout results. The memristive conductance exhibited 10 linear discrete levels from 11.1 to 100 μS. A target-aware method was adopted to program them accurately, as shown in Figure S2, Supporting Information, and the obtained stable ten-level retention is shown in Figure 3g. The conductance of the cells was controlled by the SET and RESET voltages with varying amplitudes until the programming error was within the target error ΔG. The matrix was then programmed on the 1T1R array with a maximum error of 6% and an average error of 2.8%, as shown in Figure 3h. The detailed programming data can be found in Table SII, Supporting Information.

Moreover, to investigate the correct rate of the ED engine, input voltages were applied to the memristor crossbar array via a programmed conductance to obtain the experimental MAC results. **Figure 4**a shows the MAC values for 100 temporal cycles, which represent the ED values of S and w. The tested values maintained stable fluctuations near the mean value. The original truly tested MAC results in the 1T1R array are shown in Figure S3a, Supporting Information. Owing to the differential pair for ED calculation, the standard deviation of ED results is larger than the original tested ones. But it remains two orders of magnitude smaller than the measured data which indicates the relatively stable measurement results. The analyzed test results are shown in Figure 4b. The relative error (proportion of the difference between the experimental MAC result and their mean in the latter) was a Gaussian distribution with an average value μ (≈0) and a variance σ (≈0.039), thus indicating a reliable experimental ED calculation. More simulated EDs under the

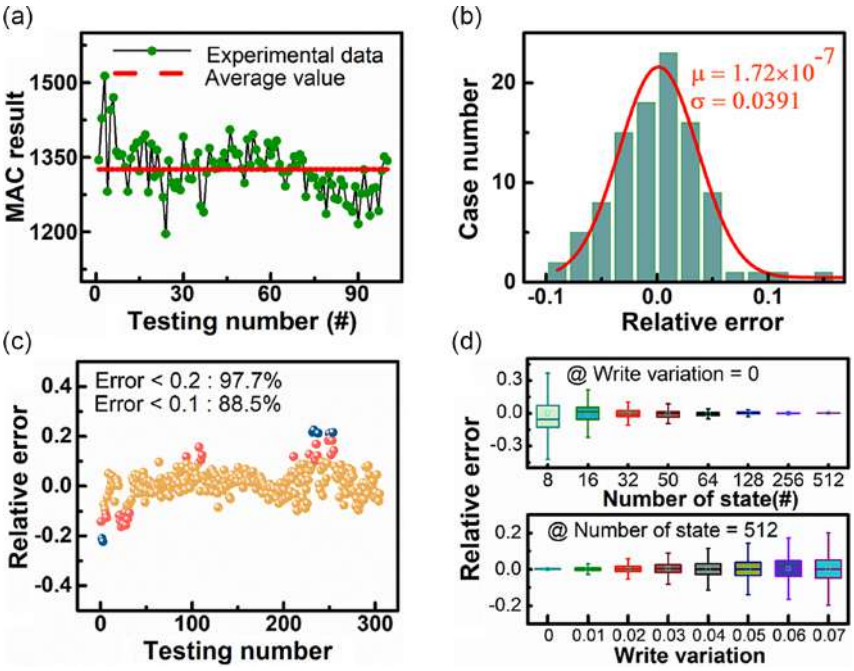

Figure 4. Functional verification of the designed memristive ED engine. a–c) The experimental results of the ED engine on the 1T1R crossbar array. a) Experimental MAC results spanning 100 temporal cycles. b) Distribution of the obtained MAC results. These are Gaussian distributed with a mean relative error μ (≈0) and a variance σ (≈0.039). c) The test results of the ED engine in different array regions show the effect of device-to-device (D2D) variation. d) The simulated tendencies of the relative errors for ED calculations as functions of increasing write variation and the number of conductance states.

© 2021 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH

26404567, 2021, 11, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/aisy.202100114 by CochraneItalia, Wiley Online Library on [30/04/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

same condition are analyzed in Figure S3b, Supporting Information, where a similar distribution of the relative errors is observed. In addition, this method performed well on the array with 88.5% errors <0.1 at different points (programming on different array regions, namely, device-to-device), as shown in Figure 4c. Furthermore, the effects of non-ideal factors of the memristor device, including the available number of conductance states and write variations, on the ED calculation were investigated via simulation (Figure 4d). The results suggest that more accurate conductance programming and smaller write–state fluctuations are beneficial to the accuracy of the calculation. Specifically, the 6-bit precision of the conductance states and 4% write variation for programming are sufficient to control the relative error of the calculation within 10%. Notably, even if the write variation is reduced to zero, or the conductance precision is as high as 10 bits, the relative error is not likely to be zero. This is because the true values (real number) can only be stored by discrete quantified conductance states. In addition, simulation results considering more factors apart from device properties are shown in Figure S3c and S3d, Supporting Information, including the input encoding noise and the stuck-at fault (SAT) of the memristor array.

2.3. Hardware Mapping of Competitive Learning Models

An in situ CLNN for clustering, as well as its optional offline inference process, is demonstrated and simulated based on the aforementioned investigation of the memristive ED engine. The control logic and data flow during the training are shown in Figure S4, Supporting Information. As shown in **Figure 5**a, the mapping rule for the two-layer competitive learning model is illustrated based on the ED engine. The input neurons represent the input pattern, and the competitive mechanism is introduced to neurons, where the output neurons of the network compete with each other and adhere to the WTA principle by measuring the EDs of the input vector and the adjustable weights. The *M*

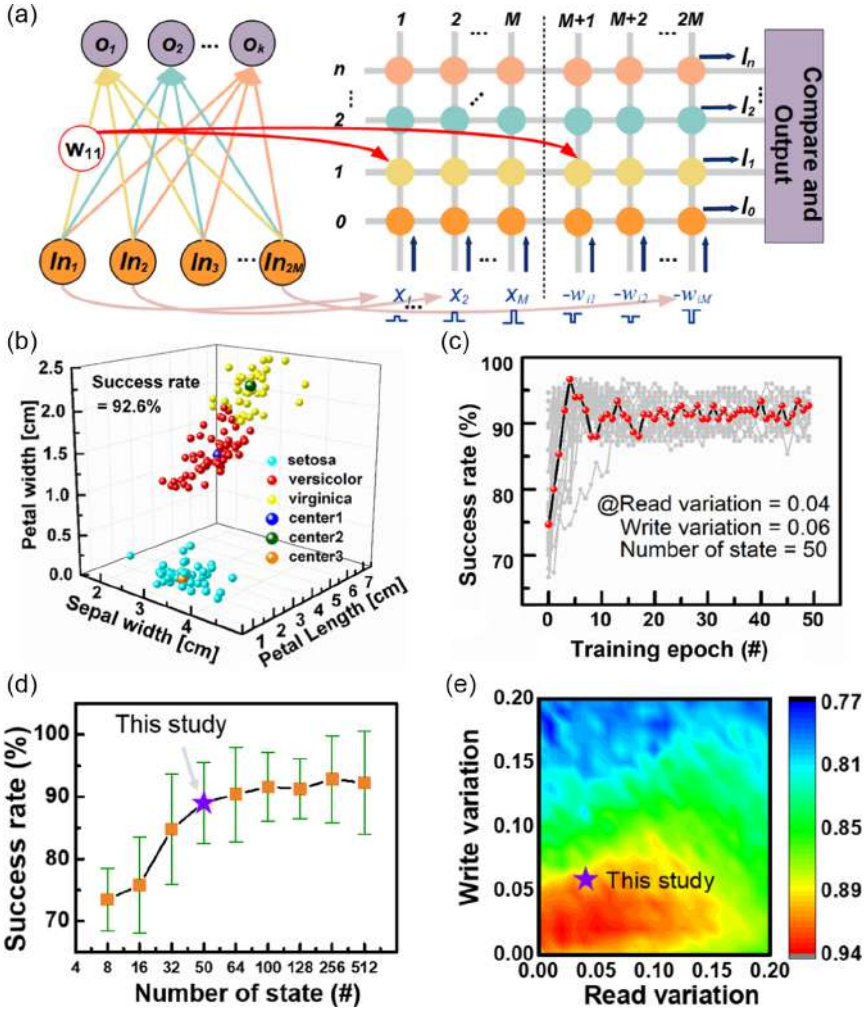

Figure 5. Simulation results for mapping CLNN to the memristor array based on the modeled experimental memristive ED engine. a) Schematic of mapping the CLNN onto the memristor array with the proposed ED engine. b) The clustering results of the IRIS dataset with a success rate of 92.6% (compared with 94% success rate in software). c) The evolution traces of convergence during the training epochs. The highlighted curve with red circles shows a typical growth process. d) The success rate distributions with different numbers of conductance states. e) The influence on success rate with varying read and write variations. In this work, 4% read noise and 6% write noise are accepted which ensures performance of the CLNN tasks.

© 2021 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH

26404567, 2021, 11, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/aisy.202100114 by CochraneItalia, Wiley Online Library on [30/04/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

weights associated with k competitive neurons were mapped to a $k \times 2M$ memristor crossbar array owing to the distributed kernels. Both the input sample vector and the weight vectors were programmed twice. The input sample vector S was stored in the 0th row. Every weight vector from the weight matrix W was programmed in the other n rows in sequence. The stored sample vector S and the ith weight vector w_i (to be compared) were then encoded as the input voltages with different amplitudes, as shown in Figure S5, Supporting Information. The readout currents I_i and I_0 were output to the external circuit to calculate the differential currents. The ith differential current represents the ED between the input vector and the ith competitive neuron. The stored sample vector S was compared with all of the weight vectors, and the output differential current was stored in a buffer sequentially until the minimum ED value was obtained by a WTA circuit outside the array. The flow chart of the training process for a two-layer CLNN is shown in Figure S6, Supporting Information. For the memristive ED engine, the calculation and comparison of a sample vector with all weight vectors were both serially computed, which was relatively time-consuming. This can be resolved by an asynchronous comparison circuit shown in Figure S7, Supporting Information, to speed up the comparison for competitive learning tasks.

In this study, prototype clustering algorithms were used as typical applications of competitive learning models based on ED calculations. The in situ training of a competitive layer learns the features to cluster different classes of inputs automatically, which exhibited the essence of unsupervised K-means. The IRIS dataset, an extensively used machine learning dataset ,[47] was adopted to verify the online training of a CLNN with an ED engine. Figure 5b shows the clustering results after training by modeling the experimental ED engine performance. The success rate of the network reached 92.6% (equivalent to 94% in software). The convergence traces are shown in Figure 5c. With only ten epochs, the success rate quickly saturated and fluctuated within a small range. The fluctuations of the success rate originated from the inevitable read and write variations, whereby the exhibited variations may skip the best weights and lead to a set of unstable trained states. This uncertainty can be improved by adaptive learning rate during training. Furthermore, the robustness of the on-chip implementation was also explored. As the conductance accuracy increased, the success rate improved simultaneously until a plateau was reached with a 6-bit accuracy (Figure 5d). The 6-bit requirement of the memristor states is rigorous for the general ED-based application. This shortage can be compensated by the device optimization or the cooperation of multiple low-precision chips. The increasing read or write variations will undoubtedly cause the collapse of the success rate. However, the write variation has a greater impact on the success rate compared with the read variation in terms of online learning (Figure 5e).

In situ training of a CLNN offers the possibility of self-adaptive application scenarios, such as autonomous driving, meaning that the weights are updated with real-time input to the net. However, for some competitive learning tasks, such as semisupervised learning vector quantization, in situ training is important, and the inference phase after training is also critical. Therefore, in **Figure 6**a, an alternative mapping design for parallel ED calculation is demonstrated due to the reconfigurability of the memristor arrays. Utilizing the training rules, the network was first trained on the simulation platform. The detailed in situ training results are shown in Figure S8, Supporting Information. After the online training, the trained weight vectors were fixed for inference. To calculate a sample vector with all of the weight matrices, the sample vector was then programmed n times to cover the left n rows of the distributed kernels that stored the trained weight vectors. Moreover, one additional column was added to store the squared $L2$ norm of the trained weights (the detailed operation processes are shown in Figure S9, Supporting Information). Herein, the breast cancer dataset,[48] which contains more samples than IRIS, was adopted as the benchmark. Figure 6b shows the clustering results of the breast cancer dataset in different situations, including software simulation, online training, and offline inference on the ED engine. Clustering results based on the ED engine were slightly lower than those obtained using software. Moreover, the result of offline inference overall yielded a higher success rate than the online training, especially for the "Malignant" class, consistent with previous publications ,[30,32] which indicates the robustness of the memristor-based inference platform. As an extension, multisample vectors could also be compared with the same weight vector, as shown in Figure S10, Supporting

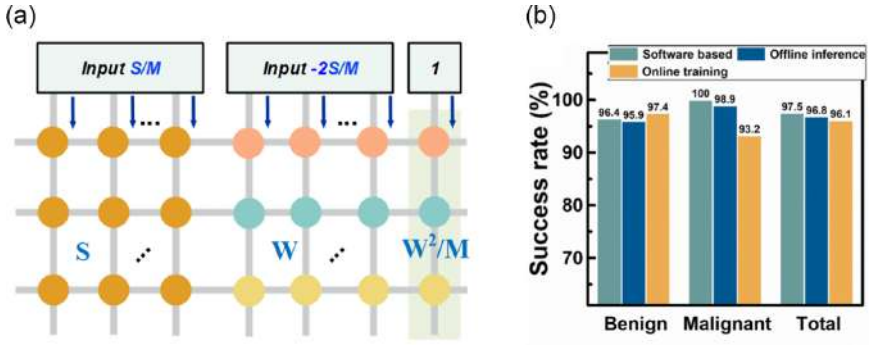

Figure 6. a) Reconfigured structure used for offline inference based on the memristive ED engine after online training. One additional row (with green background) is utilized to store the squared $L2$ norm of the trained weights W. This structure can calculate the EDs between the input vector S and the trained weights W in parallel. b) The success rates for learning vector quantization with the breast cancer dataset in different situations, including pure software simulation, offline inference, and online training on the memristive ED engine, respectively.

© 2021 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH

26404567, 2021, 11, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/aisy.202100114 by CochraneItalia, Wiley Online Library on [30/04/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Table 1. Projected energy efficiency of the inference engine based on various memristive devices, using a GPU as a benchmark ($\overline{G}$: average conductance, *T*: operational pulse width, *W*: power consumption, TOPS: Tera Operations Per Second, TOPS W^{-1}: Tera Operations Per Second Per Watt, GOPS: Giga Operations Per Second).

		$\overline{G}$	*T*	*W*/cell [μW]	Throughput	Efficiency [TOPS W^{-1}]
GPU	[49]	–	237 μs		130.5 TOPS	0.50
This study		34.4 μS	2 μs	0.545	12.29 GOPS	1.835
Projected	[50]	5.5 nS	96 ns	0.050	51.65 GOPS	84.95
	[34]	10 μS	400 ns	0.339	61.47 GOPS	181.3

Information, by storing the sample vector set on the right part of the memristor array and one weight vector on the left part repeatedly. The multichip scheme provides a matrix-to-matrix ED calculation method at the expense of a larger chip area. Utilizing the mature GPU platform as a benchmark, **Table 1** shows the projected inference efficiency of the proposed memristive ED engine using various memristive devices (the detailed calculation process is shown in the Supporting Information). In this study, the efficiency reaches 1.835 TOPS W^{-1}, which is much higher than that of a high-performance GPU (0.5 TOPS W^{-1}).[49] Potentially, utilizing state-of-the-art memristive devices, the proposed inference engine is expected to yield energy efficiency improvements that exceed 100× (181.3 TOPS W^{-1}).[34,50]

3. Conclusion

In conclusion, a fully memristive ED engine was demonstrated to compute the full expression of ED in a single-step MAC operation. Experimental verifications with 5D data were implemented on the 1T1R crossbar array, and the constant time complexity was proven regardless of the data dimensions. In situ training and offline inference schemes for the competitive learning model were developed and verified via simulated ED engines. Our results showed that the ED engine could accomplish clustering tasks with great tolerance of device variation and limited conductance states. Its performance also parallels with that of the software. Moreover, the projected energy efficiency for competitive learning exhibits a greater improvement compared with the traditional GPU. The ED engine and the memristive competitive learning models have shed light on potential edge applications by exploiting memristor-based analog computing.

4. Experimental Section

Device Fabrication and Characterization: The basic component of the 1 kb array used in this work was a hybrid integration of a metal–oxide–semiconductor field-effect transistor (MOSFET) and a TiN/HfO_x/TaO_x/TiN memristor device. The MOSFET was fabricated with a standard 0.18 μm logic process in the company SMIC, and the channel width and length were 10 and 0.35 μm, respectively. The sandwiched memristor structure was grown on the drain of the MOSFET in the following steps. The bottom TiN (40 nm) electrode was deposited on a polished *W* plug with reactive sputtering. HfO_x (10 nm) and TaO_x (50 nm) switching layers were grown by atomic layer deposition and physical vapor deposition, respectively. A 30 nm TiN layer was then deposited as the top electrode. Finally, the memristor sandwich structure was patterned using a dry etching method. The effective size of the memristor device was approximately 1 μm × 1 μm, which was defined by the etching pattern.

Electrical Measurement (the FPGA Test Board): In this study, a versatile and portable hardware platform was developed to test the resistive switching characteristics of the memristor and perform analog computing functions within the 1T1R crossbar array. The platform consisted of an FPGA-based controller, high-speed analog-to-digital and digital-to-analog converter circuits used to generate programmable pulses for reading and writing the memristors, parallel 32-channel excitation and measurement circuits for computing, independent gate voltage control circuit, two switch matrices, DDR3 circuits for data buffering, and a USB 3.0 interface to exchange data with the laptop. The platform can generate positive or negative programming pulses with a maximum amplitude of 5 V and a resolution of 10 mV. The pulse-width resolution can reach 1 ns, and the minimum rising edge is 10 ns. By utilizing the configurable feedback signal condition circuit, the weight measurement ranged from 100 Ω to 30 MΩ. The calibration algorithm was integrated into the Kintex-7 FPGA to correct the channel mismatches of the 32-channel excitation and measurement circuit, which guaranteed the accuracy of analog computing.

In Situ Training of CLNN: A two-layer competitive network was achieved with the open-source Python language (version 3.6). Some open-source libraries, including NumPy and pandas, were used to build the simulation platform. The IRIS dataset used for online training contained three classes of 50 instances each, while each instance contained four attributes: septal length and width and petal length and width. In this study, only the three most effective attributes (septal width and petal length and width) were used to verify the performance of the competitive network for online clustering tasks. During the training process, the updating rule obeyed Equation (3), and the learning rate was fixed at 0.1. The preset maximum training cycle remained at 50. This indicated that the training progress would immediately be interrupted when the 50th training cycle was reached, even though the error did not reach the target error.

Offline Inference Tasks of CLNN: The training process for offline inference verification was also implemented on the same CLNN simulation platform. The breast cancer dataset had 699 samples, 16 of which had missing values. Four hundred samples were used as the training set while the remaining were used as the testing set. Each sample had nine attributes, each of which was preprocessed to quantized values ranging from 1 to 10. The learning rate decreased or increased automatically based on the training epochs and classification results to limit the training results to its Bayesian boundary and converge. The starting learning rate was 0.3 which was recommended in the study by Kohonen et al.[51] and the learning rate would never be larger than its initial value. The training samples were randomly selected from the original training dataset.

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

H.Z., J.C., and Y. W. contributed equally to this study. This study was supported by the National Key Research and Development Plan of MOST of China (grant nos. 2019YFB2205100 and 2016YFA0203800), National Natural Science Foundation of China (grant nos. 61841404, 92064012, and 51732003), Hubei Key Laboratory of Advanced Memories, Hubei Engineering Research Center on Microelectronics, and the Chua Memristor Institute.

Conflict of Interest

The authors declare no conflict of interest.

© 2021 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH

26404567, 2021, 11, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/aisy.202100114 by CochraneItalia, Wiley Online Library on [30/04/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Data Availability Statement

Research data are not shared.

Keywords

analog computing, competitive learning, Euclidean distance engine, memristors

Received: June 7, 2021
Revised: July 8, 2021
Published online: September 7, 2021

[1] J. A. Endler, *Natural Selection in the Wild*, Princeton University Press, Princeton **1986**.
[2] D. E. Rumelhart, D. Zipser, *Cogn. Sci.* **1985**, *9*, 75.
[3] D. Kim, D. Paré, S. S. Nair, *J. Neurosci.* **2013**, *33*, 14354.
[4] X. Li, M. A. Basso, *J. Neurosci.* **2005**, *25*, 11357.
[5] H. Yang, X. Zhang, F. Yin, C. Liu, *Cvpr*, IEEE, Salt Lake City, UT **2018**, p. 3474.
[6] M. Biehl, B. Hammer, T. Villmann, *Wiley Interdiscip. Rev. Cogn. Sci.* **2016**, *7*, 92.
[7] J. Shao, F. Huang, Q. Yang, G. Luo, *IEEE Trans. Knowl. Data Eng.* **2018**, *30*, 978.
[8] E. C. Ozan, S. Kiranyaz, M. Gabbouj, *IEEE Trans. Knowl. Data Eng.* **2016**, *28*, 2884.
[9] K. L. Fair, D. R. Mendat, A. G. Andreou, C. J. Rozell, J. Romberg, D. V. Anderson, *Front. Neurosci.* **2019**, *13*, 1.
[10] S. Kaski, T. Kohonen, *Neural Networks* **1994**, *7*, 973.
[11] D. C. Park, *IEEE Trans. Neural Networks* **2000**, *11*, 520.
[12] T. Uchiyama, M. A. Arbib, *IEEE Trans. Pattern Anal. Mach. Intell.* **1994**, *16*, 1197.
[13] L. Xu, A. Krzyżak, E. Oja, *IEEE Trans. Neural Netwo.* 1993, 4, 636.
[14] S. C. Ahalt, A. K. Krishnamurthy, P. Chen, D. E. Melton, *Neural Networks* **1990**, *3*, 277.
[15] F. Lv, T. Jin, C. Yu, F. Sun, Q. Lin, K. Yang, W. Ng, in *Int. Conf. on Information and Knowledge Management Proc.*, ACM, New York, NY **2019**, p. **2635**.
[16] A. Moin, A. Zhou, A. Rahimi, A. Menon, S. Benatti, G. Alexandrov, S. Tamakloe, J. Ting, N. Yamamoto, Y. Khan, F. Burghardt, L. Benini, A. C. Arias, J. M. Rabaey, *Nat. Electron.* **2021**, *4*, 54.
[17] P. Covington, J. Adams, E. Sargin, in *RecSys 2016 - Proc. 10th ACM Conf. Recommender Systems*, ACM, New York, NY **2016**, p. 191.
[18] Z. Xu, M. Xia, *Inf. Sci. (NY).* **2011**, *181*, 2128.
[19] M. K. Najafabadi, M. N. ri Mahrin, S. Chuprat, H. M. Sarkan, *Comput. Human Behav.* **2017**, *67*, 113.
[20] L. Wang, Y. Zhang, J. Feng, *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 1334.
[21] Y. Ling, Y. An, M. Liu, S. A. Hasan, Y. Fan, X. Hu, in *Proc. Int. Jt. Conf. Neural Networks*, IEEE, Piscataway, NJ **2017**, p. 968.
[22] J. Luo, S. Wang, F. Wang, *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1986.
[23] S. Deng, L. Du, C. Li, J. Ding, H. Liu, *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2017**, *10*, 3323.
[24] D. P. P. Mesquita, J. P. P. Gomes, A. H. Souza Junior, J. S. Nobre, *Neurocomputing* **2017**, *248*, 11.
[25] O. Krestinskaya, A. P. James, L. O. Chua, *IEEE Trans. Neural Networks Learn. Syst.* **2020**, *31*, 4.
[26] W. Zhang, B. Gao, J. Tang, P. Yao, S. Yu, M. F. Chang, H. J. Yoo, H. Qian, H. Wu, *Nat. Electron.* **2020**, *3*, 371.
[27] T. Zhang, K. Yang, X. Xu, Y. Cai, Y. Yang, R. Huang, *Phys. Status Solidi – RRL* **2019**, *13*, 1.
[28] Q. Xia, J. J. Yang, *Nat. Mater.* **2019**, *18*, 309.
[29] M. Hu, C. E. Graves, C. Li, Y. Li, N. Ge, E. Montgomery, N. Davila, H. Jiang, R. S. Williams, J. J. Yang, Q. Xia, J. P. Strachan, *Adv. Mater.* **2018**, *30*, 1.
[30] 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, *Nature* **2018**, *558*, 60.
[31] Y. Zhou, H. Wu, B. Gao, W. Wu, Y. Xi, P. Yao, S. Zhang, Q. Zhang, H. Qian, *Adv. Funct. Mater.* **2019**, *29*, 1.
[32] F. Alibart, E. Zamanidoost, D. B. Strukov, *Nat. Commun.* **2013**, *4*, 1.
[33] L. Gao, P. Y. Chen, S. Yu, *IEEE Electron Device Lett.* **2016**, *37*, 870.
[34] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, H. Qian, *Nature* **2020**, *577*, 641.
[35] S. Yu, in *Proc.—IEEE Int. Symp. Circuits Systems*, IEEE, Piscataway, NJ **2014**, p. 1058.
[36] Y. J. Jeong, J. Lee, J. Moon, J. H. Shin, W. D. Lu, *Nano Lett.* **2018**, *18*, 4447.
[37] P. M. Sheridan, F. Cai, C. Du, W. Ma, Z. Zhang, W. D. Lu, *Nat. Nanotechnol.* **2017**, *12*, 784.
[38] Y. Jiang, J. Kang, X. Wang, *Sci. Rep.* **2017**, *7*, 1.
[39] Z. Sun, G. Pedretti, E. Ambrosi, A. Bricalli, W. Wang, D. Ielmini, *Proc. Natl. Acad. Sci. U. S. A.* **2019**, *116*, 4123.
[40] F. M. Bayat, M. Prezioso, B. Chakrabarti, H. Nili, I. Kataeva, D. Strukov, *Nat. Commun.* **2018**, *9*, 1.
[41] Q. Liu, B. Gao, P. Yao, D. Wu, J. Chen, Y. Pang, W. Zhang, Y. Liao, C. Xue, W. Chen, J. Tang, Y. Wang, M. Chang, H. Qian, H. Wu, in *2020 IEEE Int. Solid- State Circuits Conf.*, IEEE, Piscataway, NJ **2020**, p. 500.
[42] S. Amari, S. Wu, *Neural Networks* **1999**, *12*, 783.
[43] H. Wu, C. Wang, N. F. Tzeng, *IEEE/ACM Trans. Netw.* **2005**, *13*, 609.
[44] C. H. Li, C. T. Lin, B. C. Kuo, H. H. Ho, in *Proc.—Int. Conf. Technol. Application on Artificial Intelligence*, IEEE, Piscataway, NJ **2010**, p. 226.
[45] B. Kosko, *IEEE Trans. Neural Networks* **1991**, *2*, 522.
[46] B. D. Brown, H. C. Card, *IEEE Trans. Comput.* **2001**, *50*, 906.
[47] D. Dua, C. Graff, *UCI Machine Learning Repository*, University of California, School of Information and Computer Science, Irvine, CA **2019**, http://archive.ics.uci.edu/ml.
[48] K. P. Bennett, O. L. Mangasarian, *Optim. Methods Softw.* **1992**, *1*, 23.
[49] P. Lin, C. Li, Z. Wang, Y. Li, H. Jiang, W. Song, M. Rao, Y. Zhuo, N. K. Upadhyay, M. Barnell, Q. Wu, J. J. Yang, Q. Xia, *Nat. Electron.* **2020**, *3*, 225.
[50] R. Berdan, T. Marukame, K. Ota, M. Yamaguchi, M. Saitoh, S. Fujii, J. Deguchi, Y. Nishi, *Nat. Electron.* **2020**, *3*, 259.
[51] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, K. Torkkola, *Learning* **1996**, *30*, 10625.

© 2021 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH

26404567, 2021, 11, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/aisy.202100114 by CochraneItalia, Wiley Online Library on [30/04/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License