

Brain-inspired memristive pooling method for enhanced edge computing

Wenbin Guo(郭文斌), Zhe Feng (冯哲), Haochen Wang (王昊辰), Zhihao Lin(蔺志豪), Jianxun Zou(邹建勋), Zuyu Xu(徐祖雨), Yunlai Zhu(朱云来), Yuehua Dai (代月花), and Zuheng Wu (吴祖恒)
Citation: Chin. Phys. B, 2025, 34 (**12**): 127301. DOI: 10.1088/1674-1056/adfefb
http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

Exploring reservoir computing: Implementation via double stochastic nanowire networks

Jian-Feng Tang(唐健峰), Lei Xia(夏磊), Guang-Li Li(李广隶), Jun Fu(付军), Shukai Duan(段书凯), and Lidan Wang(王丽丹)
Chin. Phys. B, 2024, 33 (**3**): 037302. DOI: 10.1088/1674-1056/aceeea

Simulation of optical and electrical synaptic functions in MoS_2/α-In_2Se_3 heterojunction memtransistors

Tao Xiang(相韬), Fengxiang Chen(陈凤翔), Xiaoli Li(李晓莉),Xiaodong Wang(王小东), Yuling Yan(闫誉玲), and Lisheng Wang(汪礼胜)
Chin. Phys. B, 2023, 32 (**11**): 117301. DOI: 10.1088/1674-1056/acee58

High-performance artificial neurons based on Ag/MXene/GST/Pt threshold switching memristors

Xiao-Juan Lian(连晓娟), Jin-Ke Fu(付金科), Zhi-Xuan Gao(高志瑄),Shi-Pu Gu(顾世浦), and Lei Wang(王磊)
Chin. Phys. B, 2023, 32 (**1**): 017304. DOI: 10.1088/1674-1056/ac673f

Digital and analog memory devices based on 2D layered MPS_3 (M=Mn, Co, Ni) materials

Guihua Zhao(赵贵华), Li Wang(王力), Xi Ke(柯曦), and Zhiyi Yu(虞志益)
Chin. Phys. B, 2021, 30 (**4**): 047303. DOI: 10.1088/1674-1056/abd397

Implementation of synaptic learning rules by TaO_x memristors embedded with silver nanoparticles

Yue Ning(宁玥), Yunfeng Lai(赖云锋), Jiandong Wan(万建栋), Shuying Cheng(程树英), Qiao Zheng(郑巧), and Jinling Yu(俞金玲)
Chin. Phys. B, 2021, 30 (**4**): 047301. DOI: 10.1088/1674-1056/abccb8

SPECIAL TOPIC — Biophysical circuits: Modeling & applications in neuroscience

Brain-inspired memristive pooling method for enhanced edge computing

Wenbin Guo(郭文斌)[†], Zhe Feng (冯哲)[†], Haochen Wang (王昊辰), Zhihao Lin(蔺志豪), Jianxun Zou(邹建勋), Zuyu Xu(徐祖雨), Yunlai Zhu(朱云来)[‡], Yuehua Dai (代月花)[§], and Zuheng Wu (吴祖恒)[¶]

School of Integrated Circuits, Anhui University, Hefei 230601, *China*

(Received 24 June 2025; revised manuscript received 25 August 2025; accepted manuscript online 26 August 2025)

Edge deployment solutions based on convolutional neural networks (CNNs) have garnered significant attention because of their potential applications. However, traditional CNNs rely on pooling to reduce the feature size, leading to substantial information loss and reduced network robustness. Herein, we propose a more robust adaptive pooling network (APN) method implemented using memristor technology. Our method introduces an improved pooling layer that reduces input features to an arbitrary scale without compromising their importance. Different coupling coefficients of the pooling layer are stored as conductance values in arrays. We validate the proposed APN on generic datasets, demonstrating significant performance improvements over previously reported CNN architectures. Additionally, we evaluate the APN on a CAPTCHA recognition task with perturbations to assess network robustness. The results show that the APN achieves 92.6% accuracy in 4-digit CAPTCHA recognition and exhibits higher robustness. This brief presents a highly robust and novel scheme for edge computing using memristor technology.

Keywords: memristor, edge computing, adaptive pooling, image classification

PACS: 73.40.Sx, 84.35.+i, 07.05.Pj, 87.19.lv

DOI: 10.1088/1674-1056/adfefb **CSTR:** 32038.14.CPB.adfefb

1. Introduction

Convolutional neural networks (CNNs) are designed to mimic the functions of the human brain through interconnected neurons, allowing them to handle complex nonlinear systems.[1] CNNs have been widely used in computer vision (CV),[2] natural language processing (NLP),[3] and speech recognition.[4] However, the explosive development of edge computing technology has increased the demand for processing speed in computers and edge devices. Owing to the separation of processing units and memory, traditional complementary metal–oxide–semiconductor (CMOS) hardware platforms with von Neumann architecture face obstacles in meeting these growing computational demands.[5] To meet the increasing demand for computational power, developing new computational paradigms for efficient hardware neuromorphic computing is a promising direction.

Memristors are considered one of the most promising devices for overcoming computational bottlenecks. These devices can perform both memory and processing functions, promoting the concept of computing-in-memory (CIM). Memristors have also attracted significant attention owing to their low power consumption and high density in neuromorphic applications, including neural networks.[6–13] Among many tasks, the combination of memristors and CNNs has proven advantageous in terms of high parallelism and energy efficiency.[14] However, previous work has mainly focused on the physical deployment and hardware acceleration of CNNs.[15] A major issue in implementing edge computing is the poor network robustness and limited tolerance to perturbations.[16] In CNNs, pooling layers reduce feature size and preserve key information after convolution but lack a biological basis. These layers, such as average and max pooling, cause significant information loss.[17] Max pooling can lead to overfitting and does not ensure generalization, whereas average pooling suppresses important activation values by considering only average information.[18] Both types reduce CNNs' ability to identify spatial relationships, decreasing robustness.

Recently, pooling-related methods have received increased attention, and several solutions have been proposed. Some approaches attempt to replace pooling layers using large-stride convolutional parameters or transformer architectures. Although transformers perform well without pooling, they involve complex parameters and are difficult to deploy in hardware. Similarly, using large-stride convolutions increases parameter count and computational complexity. New techniques, such as local importance pooling and continuous-time

[†]These authors contributed equally to this work.
[‡]Corresponding author. E-mail: zhuyunlai@ahu.edu.cn
[§]Corresponding author. E-mail: daiyuehua@ahu.edu.cn
[¶]Corresponding author. E-mail: wuzuheng@ahu.edu.cn

© 2025 Chinese Physical Society and IOP Publishing Ltd. All rights, including for text and data mining, AI training, and similar technologies, are reserved.

pooling,[19,20] aim to optimize these issues but still face challenges owing to additional training and computational load. Novel learning-based pooling methods, such as the end-to-end learning-based pooling (LEAP) architecture, introduce extra backpropagation operations,[21] further complicating the system. Thus, achieving efficient learning by considering the interrelationships among features remains a significant challenge.

In this work, inspired by the biological self-feedback loop mechanism (SLM), we implement an adaptive pooling network (APN) using memristor arrays. To enhance bio-interpretability, we simulate cortical column functions with feature columns. Unlike traditional networks, our pooling layer is placed only before the fully connected layer. We tested the APN on generic datasets, and it significantly outperformed existing CNNs in five image classification tasks. Using a CAPTCHA dataset with random rotations and perturbations, the APN demonstrated superior robustness, achieving 89.6% accuracy with a 50% dropout rate. This approach is suitable for deployment on edge devices requiring high robustness for complex tasks.

2. Concepts of memristor-based AP method

This study proposes an adaptive pooling (AP) method inspired by biological systems, where learning and memory occur through neural activity. As shown in Fig. 1, in visual tasks, the retina converts light into electrical signals, which are processed by the brain's primary visual cortex. Cortical columns, considered basic units of the neocortex,[22] encode visual features such as color, shape, and movement. Unlike CNN feature maps, cortical columns hold more complex information. The brain uses a SLM to adjust synaptic weights for learning. The SLM was recently discovered in the mammalian brain.[23] Input signals are received by motor-cortex-projecting (MP) neurons. These MP neurons create a relatively isolated local self-feedback loop for learning the interrelationships among features by regulating synaptic weights, which inspired this work. Although its mechanisms are still under further debate and investigation, its bio-inspired properties can be exploited in certain machine learning algorithms. Inspired by this learning mechanism, the APN, shown in the lower part of Fig. 1, extracts features via convolutional layers, forming feature columns that encapsulate abstract information such as shape, position, and rotation angle. These columns are represented as three-dimensional (3D) tensors with shared weights to recognize the existence of visual objects and encode their attributes. The update process will be discussed in the following section. After training the coupling coefficients, the memristor-based APN simulates self-feedback learning. The output is then used in the fully connected layer for classification and recognition tasks.

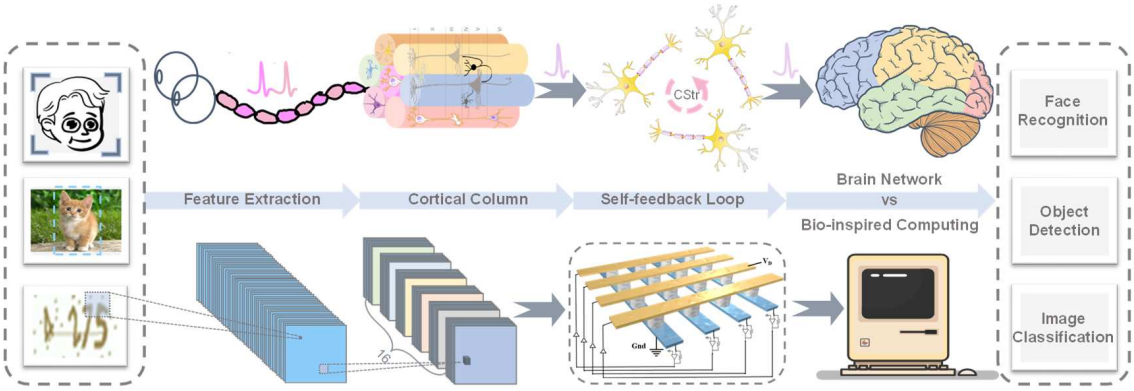


Fig. 1. The application processes of biological neural systems and bio-inspired computing systems for vision tasks, including face recognition, object detection, and image recognition.

3. Implementation of memristor-based APN

This section first introduces the original APN algorithm framework, then explains the algorithm simplification and proof process, and finally presents the APN method proposed in this paper along with the corresponding figure. Standard CNNs can generate powerful features, but they often neglect modeling similar features and lack the ability to cluster location-aware features. The APN aims to learn feature interrelationships, building on Sabour's framework.[24] Firstly, we introduce the framework of Sabour *et al*. and propose an optimization method. In Sabour's framework, encapsulated data are expanded into vectors, serving as input $u_{j|i}$. The target output s_j aggregates all data $u_{j|i}$ using coupling coefficients c_{ij}

$$s_j = \sum_{0}^{i} c_{ij} u_{j|i}, \tag{1}$$

where s_j represents the feature aggregation result and c_{ij} indicates the coupling coefficient of the i-th vector and j-th vector. Initially, all c_{ij} are equal, summing to 1. To ensure that the length of the output vector is less than 1, s_j is squeezed using

a squeezing function

$$v_j = \frac{\|s_j\|^2}{1+\|s_j\|^2}\frac{s_j}{\|s_j\|}, \tag{2}$$

where v_j is the j-th output vector and s_j is its input vector. Equation (2) guarantees that the output vector direction remains constant and that its probability lies between 0 and 1, ensuring the stability of the training process.

In this work, to deploy the proposed APN scheme using memristor technology, we further integrate Eqs. (1) and (2) from the previous framework into

$$v_j = \sum_0^i c_{ij} u_{j|i}. \tag{3}$$

This is because, when using memristor arrays for computation, normalization mapping is required to resolve the dimensional mismatch between software and hardware, which naturally ensures the convergence of the output vector and avoids introducing the aggregation results s_j. The specific derivation process is as follows.

During inference, to calculate the output vector v_j, the coupling coefficient c_{ij} is mapped to the conductance value G_{ij} of the hardware memristor array, and the input information $u_{j|i}$ is mapped to the corresponding read voltage V_{ij}. Efficient computation is then achieved through the intrinsic vector-matrix multiplication (VMM) of the array. Specifically, the input $u_{j|i}$ converted to the corresponding read voltage V_{ij} through normalization can be expressed as

$$\frac{u_{j|i}-u_{j|\min}}{u_{j|\max}-u_{j|\min}} = \frac{V_{ij}-V_{\min,j}}{V_{\max,j}-V_{\min,j}}, \tag{4}$$

where Eq. (4) represents the normalized mapping equation used to convert software parameter input $u_{j|i}$ into hardware read voltage V_{ij}. The term $V_{\max,j}-V_{\min,j}$ is the difference between the maximum ($V_{\max,j}=0.3$ V) and minimum ($V_{\min,j}=0$ V) read voltage. Similarly, $u_{j|\max}-u_{j|\min}$ is the difference between the maximum ($u_{j|\max}$) and minimum ($u_{j|\min}$) input values $u_{j|i}$. After transformation

$$V_{ij} = \frac{V_{\max,j}-V_{\min,j}}{u_{j|\max}-u_{j|\min}}(u_{j|i}-u_{j|\min}) \le V_{\max}. \tag{5}$$

Similar to the mapping of inputs to read voltages, the coupling coefficients must also be mapped to conductance values in the array. Specifically, the coupling coefficient c_{ij} expressed by conductance G_{ij} can be written as

$$G_{ij} = \frac{G_{\max,j}-G_{\min,j}}{c_{\max,j}-c_{\min,j}}(c_{ij}-c_{\min,j})+G_{\min,j} \le G_{\max,j}, \tag{6}$$

where Eq. (6) represents the normalized mapping formula used to convert the coupling coefficient c_{ij} into conductance G_{ij}. The term $G_{\max,j}-G_{\min,j}$ is the difference between the maximum ($G_{\max,j}$) and minimum ($G_{\min,j}$) conductance. Likewise, $c_{\max,j}-c_{\min,j}$ is the difference between the maximum ($c_{\max,j}$) and minimum ($c_{\min,j}$) coupling coefficients. The output current I_j corresponding to the column output v_j in the memristor array can then be expressed as

$$I_j = \sum_0^i G_{ij}V_{ij} \le \sum_0^i G_{\max,j}V_{\max,j}, \tag{7}$$

which represents the j-th output vector v_j in software. From Eq. (7), we deduce

$$\frac{G_{ij}V_{ij}}{G_{\max,j}V_{\max,j}} \le 1. \tag{8}$$

This ensures the convergence of v_j during the coupling coefficient update process.

The update process then involves adjusting the coupling coefficients

$$b_{ij} = b_{ij} + u_{j|i} \times v_j, \tag{9}$$

$$c_{ij} = \mathrm{Softmax}(b_{ij}). \tag{10}$$

Here, b_{ij} is an intermediate variable for calculating coupling coefficients, initially set to 0. The Softmax function assigns probabilities to b_{ij}, determining the importance of the data in each column. The updated equations for the optimized algorithm are shown in Fig. 2(a).

By multiplying and adding the input data $u_{j|i}$ and the output vector v_j, larger results indicate greater importance of the data in that column. A higher probability is therefore assigned in the next iteration. Consequently, depending on the different input feature data, the pooling coefficients are learned differently, achieving AP. With the feedback loops established by the above rules, it is possible to build adjustable learning mechanisms for the importance of input features and to characterize importance levels using coupling coefficients c_{ij}, where more important features have larger c_{ij} in the feedback loop.

Initially, all c_{ij} values are $1/i$, corresponding to average pooling. As iterations increase, most c_{ij} become 0, with only a few remaining nonzero, thereby realizing max pooling. The APN allows learning hidden information among feature data, enhancing network robustness. Additionally, the coupling coefficients can be adjusted based on input $u_{j|i}$ and output s_j requirements. This allows a single layer before fully connected layers to match the size of multiple convolution-pooling pairs without adjusting convolution stride parameters. By avoiding multiple AP layers, the update cost is reduced and performance degradation from overfitting is prevented.

Next, we propose a memristor-based APN methodology using a memristor array topology. Hardware deployment of the algorithm can enhance application scenarios in edge computing. The array topology enables rapid inference with a 20 ns read pulse, accelerating the algorithm and consuming little power. This system can serve as an edge device, handling

complex tasks while training occurs in the cloud. As shown in Fig. 2(b), $u_{j|i}$ is mapped as the read voltage to the input buffer and then applied to the array, where column-wise matrix addition is performed. Memristor array weights represent the coupling coefficients c_{ij}, and a differential circuit handles positive and negative feature data operations during inference. Also, Fig. 2(c) illustrates the circuit schematic of the training process. The control signal is applied to the control circuit through transmission gates, and the output vector v_j is scaled by trans-impedance amplifiers (TIAs) before being sent to the backend for VMM and coefficient updates. After nonlinear activation, the pulse generation module updates the coupling coefficient matrix, continuing until the specified number of iterations is reached. Detailed measurements of the memristor and peripheral circuits are available in previous research.[25]

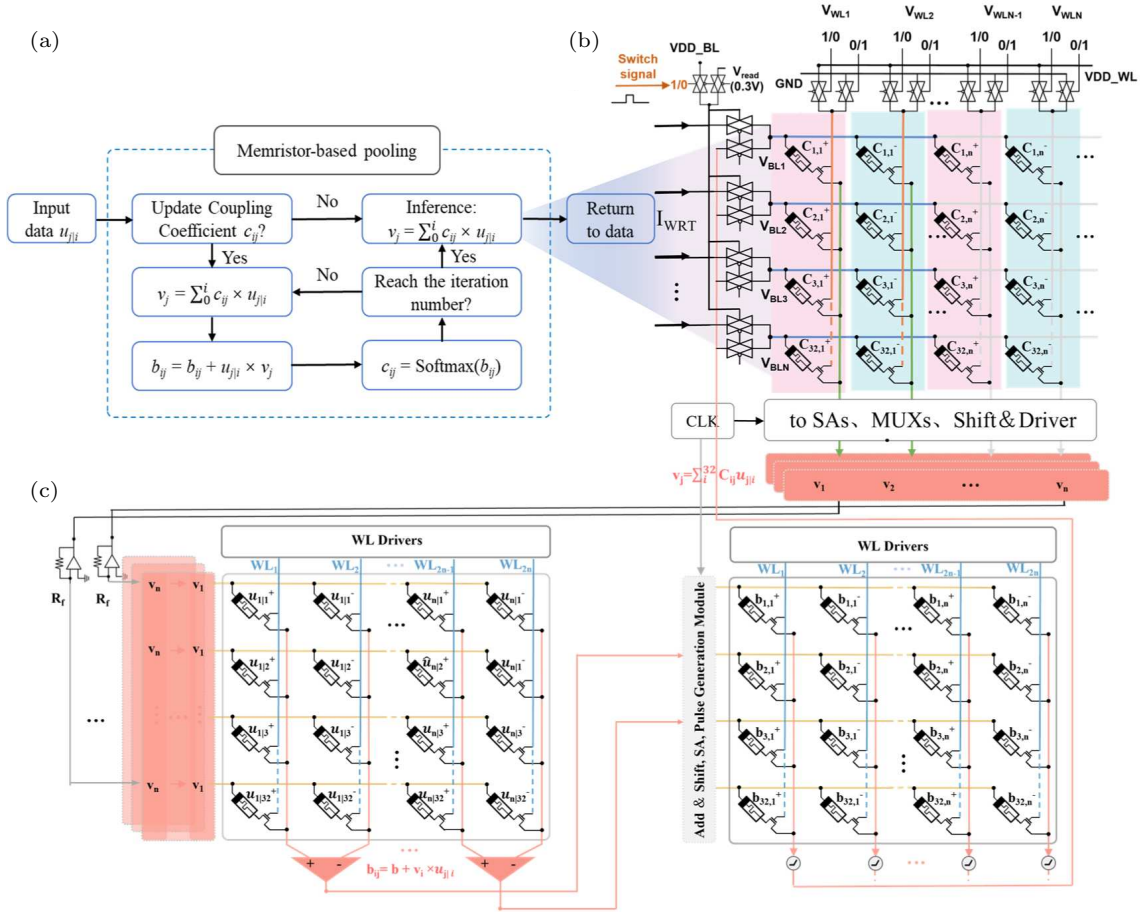


Fig. 2. (a) The flowchart of the updating process of the memristor-based APN. (b) The circuit schematic for the inference process of the memristor-based APN. (c) The circuit schematic for the training process of the memristor-based APN.

4. Experiments and results

In the validation process, the performance of the proposed APN is first verified on generic image classification datasets. Firstly, a fashion Modified National Institute of Standards and Technology (MNIST) dataset recognition task was validated using a convolutional layer and an AP layer for feature learning, followed by a fully connected layer for recognition. For comparison, a CNN model with two convolutional–maximum pooling pairs (32 and 48 channels) was used, feeding the output into a fully connected network for classification. Figure 3(a) shows the results over 50 epochs, with test accuracies of 95.95% (APN) versus 91.75% (CNN). The quick convergence of the CNN model may arise from the property of extracting important features from maximum pooling layers, but the neglect of other feature influences prevents the model from learning sufficient effective information to achieve higher performance. The system performance was further evaluated against reported CNN models using the MNIST, street view house number (SVHN), Cifar10, and Cifar100 datasets. The multiple test results are shown in Fig. 3(b), where the proposed APN exhibits significant performance improvement compared to reported CNN models. Notably, on the Cifar100 dataset, the model with two convolutional layers and one AP layer achieved 65.2% accuracy, 8% higher than the AlexNet-C100 structure.[28] Notably, the APN has fewer network layers and parameters (fewer than two convolutional-AP layers) while achieving higher performance benchmarks, indicating that the network has excellent feature extraction and learning capability, enabling superior performance with a simpler structure. Table 1 lists the relevant details for testing on different

datasets. Hardware configurations include an Intel i5-13490F and an NVIDIA 2060Ti graphics card. Neural network simulations were conducted using Python and the PyTorch library. For the MNIST, SVHN, Fashion MNIST, and CIFAR-10 tasks, training was performed using a single convolutional layer followed by an AP layer and a single fully connected layer. During simulation, the batch size was set to 128, the learning rate was 0.001, and the learning rate decay was 0.96. The number of iterations was 1, 1, 1, and 3, respectively. In contrast, the comparison algorithms for the MNIST and Fashion MNIST tasks use two convolutional layers with max pooling, a batch size of 128, and a learning rate decay of 0.9. Compared to traditional schemes, the proposed APN demonstrates a significant performance improvement.

The robustness and performance of the memristor-based APN system are further evaluated using a digital CAPTCHA-based recognition task, as shown in Fig. 4. The CAPTCHA dataset, first proposed by von Ahn,[29] is widely considered a security measure to determine whether a user is human or a machine. Each CAPTCHA image features an RGB channel with four digits, each randomly positioned and rotated within a 45-degree range. In the CAPTCHA recognition task, a two-level branching structure extracts multiple feature levels, which are combined into feature blocks sequentially with short connections to enhance information learning. After coupling with the network weights, the feature block data is flattened into the 32×32 memristor array for the AP process. For the APN, when the iteration number is 0, according to the algorithm update rule $c_{ij} = \mathrm{Softmax}(b_{ij})$, the conductance of the pooling coefficient matrix c_{ij} becomes identical, which behaves as average pooling.

Table 1. Comparison of one-epoch average training time for different algorithms (APN and CNN) on different datasets.

Model	APN		CNN (Baseline)	
Dataset	Structure	Train time	Structure	Train time
MNIST	Conv + AP	7.1 s	2Conv + 2Max_pool	6.1 s
SVHN	Conv + AP	19.6 s	GoogleNet[26]	–
Fashion MNIST	Conv + AP	7.4 s	2Conv + 2Max_pool	6.14 s
Cifar10	Conv + AP	21.7 s	VGG16[27]	–
Cifar100	2Conv + AP	25.3 s	AlexNet-C100[28]	–

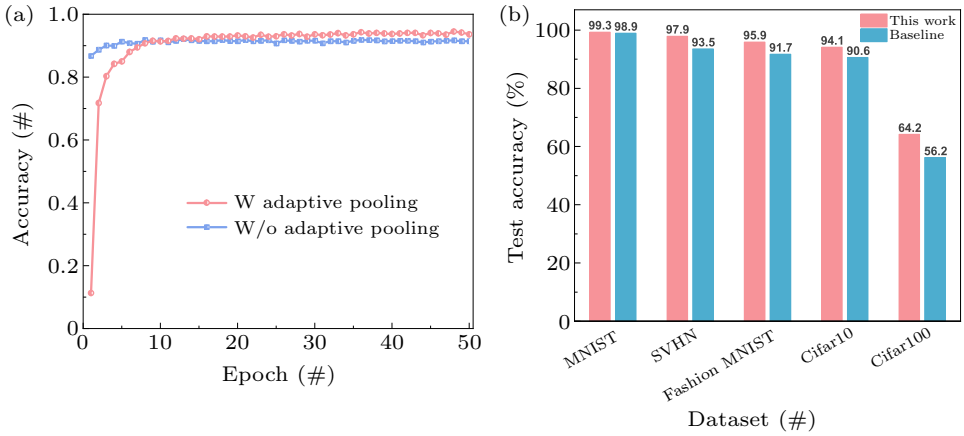


Fig. 3. (a) Comparison of test results on the generic Fashion MNIST dataset recognition task, where the blue line represents the generic CNN network and the red line represents the proposed APN in this work. (b) Test results of the APN (red) and CNN algorithms (aqua-green) on different datasets. Comparison results for the SVHN, Cifar10, and Cifar100 datasets are taken from Refs. [26–28], respectively.

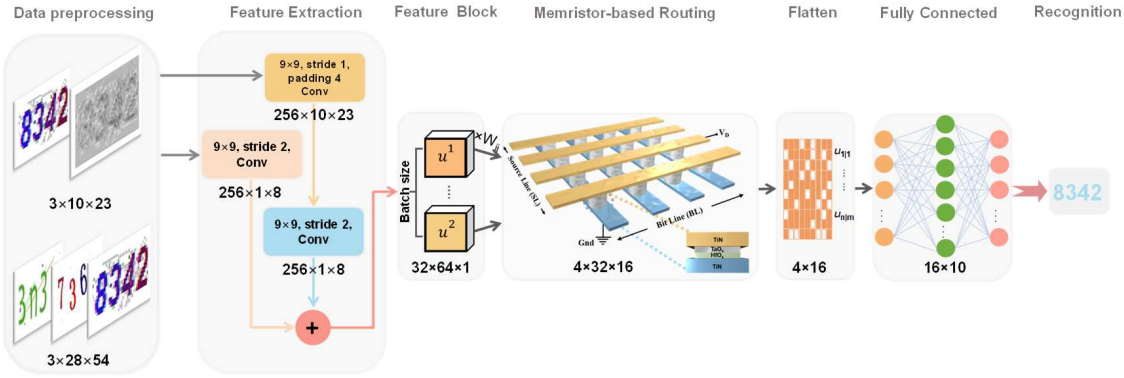


Fig. 4. The running process of the APN applied to the CAPTCHA recognition task.

After training, the coupling coefficients are first mapped to array conductance values as shown in Fig. 5(a), using the normalization computational rules of Eq. (6). The initial conductance distribution is displayed in the left panel of Fig. 5(a), obtained by applying a uniform programming voltage to the array. As shown in Fig. 5(b), the average conductance is about

33 μS, indicating a quasi-normal distribution of conductance in the memristor cells. This distribution can be further optimized by adjusting the electrical operation parameters, enabling flexible hardware implementation of the memristor-based APN. As shown in the right panel of Fig. 5(a), after three iterations, the corresponding correlations of the AP coefficient matrix are successfully learned for the input features.

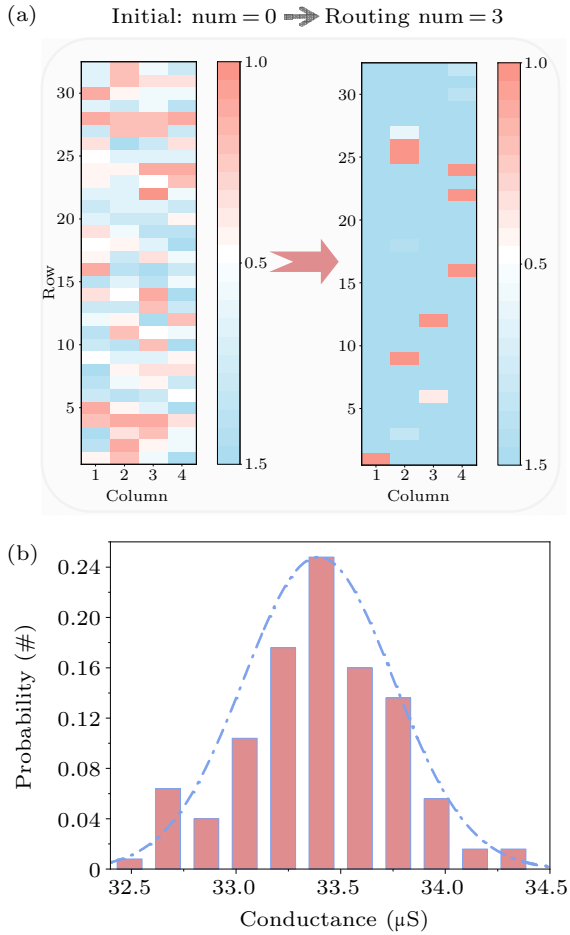


Fig. 5. (a) The initial normalized weight distribution and the normalized weight distributions after three iterations in the memristor array. (b) The initial weight distribution of the memristor array.

The CAPTCHA recognition task-based APN was first simulated and analyzed in software, where the memristor weight nonlinearity (NL) was modeled using Python. A CNN model with the same structure (using maximum pooling layers) was used as a baseline (blue line) for performance comparison. To determine the system's highest achievable performance, the iterations corresponding to the highest accuracy over 50 epochs were evaluated. The inset of Fig. 6(a) shows the test accuracy for iteration numbers from 1 to 10, indicating that the system performs best with three iterations, achieving an accuracy of about 94.5%. Although accuracy may slightly improve with increasing iteration numbers, the corresponding power consumption increases accordingly. The robustness of the APN architecture was assessed before mapping to the memristor array, as shown in Fig. 6(b). Non-ideal array factors and peripheral circuit conversion inaccuracies may introduce fluctuations in coupling coefficients during mapping and inference, potentially degrading accuracy. Figure 6(b) shows that even with noise levels up to 30% applied to the coupling coefficients during inference, the network accuracy remains above 85%. Figure 6(c) demonstrates that adding a dropout operation to emulate stuck-at-zero (SA0) memristor failures during array mapping shows that the APN maintains over 85% accuracy at a 50% dropout ratio. In contrast, CNN accuracy without AP decreases almost linearly as the dropout ratio increases, highlighting the advantage of the proposed system.

During the experiment, image data is processed through convolution and then sequentially converted into voltage vectors via a digital-to-analog converter (DAC) and analog switch arrays. These voltages are normalized and then fed into the memristor array for computation. The output current is converted and returned to the computer via an analog-to-digital converter (ADC) for prediction. In principle, by further deploying the network layer weights, the data transmission overhead can be further reduced, enabling full in-hardware deployment of the network layers. Offline learning was used to map the pre-trained coupling coefficients with an iteration number of 3, and the experimental results after 50 training epochs are illustrated in Fig. 6(d). The hardware experiment achieved a final accuracy of 92.6%, slightly below the 94.5% achieved in simulations but still 3.3% higher than the baseline CNN's 89.3%. Notably, in repeated tests, CNN-based models frequently fail to converge due to issues arising during early-stage training. In contrast, the APN consistently converges regardless of initial conditions, demonstrating superior robustness and stability. This difference may stem from the inherent instability and poor robustness of CNN models during training.

During inference, the memristor array used for VMM operations is driven by a maximum read voltage pulse of 0.3 V with a duration of 20 ns (note that the 20 ns read pulse is significantly longer than the worst-case RC time constant). Therefore, the maximum power consumption for each inference sample is $E_{\text{array}} = \sum u^2 tG = u^2 \times 20\ \text{ns} \times 33 \times 10^{-6} S \times 4 \times 32 \times 16 = 0.09 \times 20 \times 10^{-9} \times 33 \times 10^{-6} \times 4 \times 32 \times 16 \approx 121.65$ pJ, and the execution time of the array is $T_{\text{array}} = 4 \times 32 \times 16 \times 20\ \text{ns} \approx 40.9$ μs. In addition, the DAC power consumption varies significantly depending on its type and fabrication process. Here, considering the use of a single DAC,[30] the maximum intrinsic energy cost for completing one inference is estimated as $E_{\text{DAC}} = 12.6\ \text{mW} \times 5\ \text{ns} \times 8 \times 4 \times 32 \times 16 \approx 1.03$ μJ (0.18 μm, 8-bit, 200 MHz). Similarly, $T_{\text{DAC}} = 5\ \text{ns} \times 8 \times 4 \times 32 \times 16 \approx 81.9$ μs (the data size processed through the memristor array is $4 \times 32 \times 16$).

In summary, this section validates the superior performance and robustness of the proposed memristor-based APN across both generic image classification tasks and a

complex CAPTCHA recognition scenario with perturbations. Hardware-aware simulations and analysis confirm its potential for low-power, high-speed edge computing. However, several open challenges must be addressed for future large-scale deployment, including mitigating the effects of device nonlinearity and variations on learning accuracy, developing efficient strategies for in-situ training directly on memristor arrays to bridge the software–hardware gap, and co-optimizing the algorithm and peripheral circuits (e.g., DAC/ADC) for improved energy efficiency and throughput.

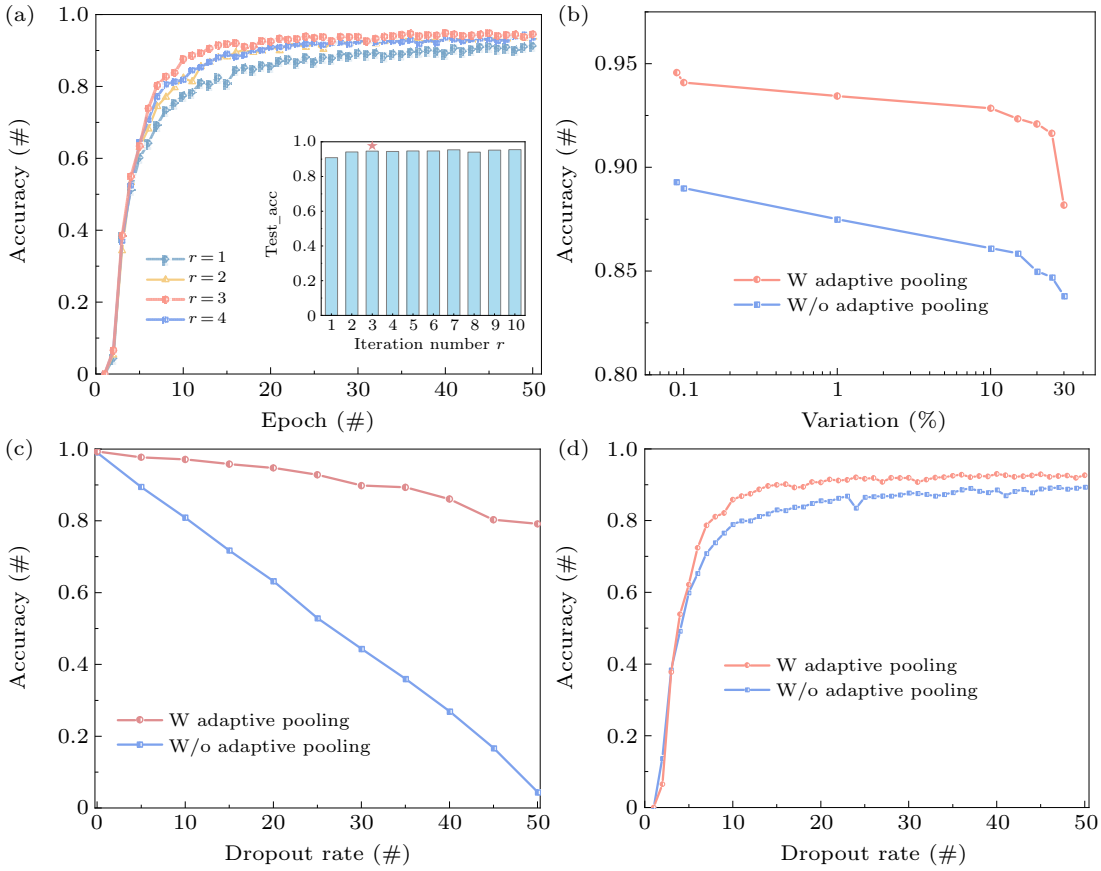


Fig. 6. (a) Test accuracy corresponding to different iterations at 50 epochs. The inset shows the network test accuracy from 1 to 10 iterations. (b) Test accuracy corresponding to different variation rates. (c) Training accuracy of the networks with (APN model) and without (CNN model) adaptive pooling at different dropout rates. (d) Test accuracy of the networks with and without adaptive pooling (CNN model) for the CAPTCHA recognition task under 50 epochs.

5. Conclusion

In summary, inspired by biological brain learning mechanisms, this work introduces an optimized adaptive pooling method. To enable hardware deployment of the algorithm, the pooling coefficients are replaced using memristor arrays during inference, achieving a highly robust and high-performance system. In future work, the system performance can be further improved by optimizing the nonlinearity of the weight updates as well as mitigating variation effects and other non-ideal factors in the memristor devices. This work presents a novel solution for implementing memristor-based edge computing.

Acknowledgments

Project supported by the National Natural Science Foundation of China (Grant Nos. 62274002, 62304001, and 62201005), the Anhui Provincial Natural Science Foundation (Grant Nos. 2308085QF213 and 2408085QF211), and the Natural Science Research Project of the Anhui Educational Committee (Grant No. 2023AH050072).

References

[1] Abiodun O I, Jantan A, Omolara A E, Dada K V, Mohamed N A and Arshad H 2018 *Heliyon* **4** e00938
[2] Szegedy C, Vanhoucke V, Ioffe S, Shlens J and Wojna Z 2016 *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)* pps. 2818–2826
[3] Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, Liu T, Wang X, Wang G, Cai J and Chen T 2018 *Pattern Recognit.* **77** 354
[4] Abdel-Hamid O, Mohamed A R, Jiang H, Deng L, Penn G and Yu D 2014 *IEEEACM Trans. Audio Speech Lang. Process.* **22** 1533
[5] Kundu S, Ganganaik P B, Louis J, Chalamalasetty H and Rao B P 2022 *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **30** 755
[6] Dai Y, Guo W, Feng Z, Xu Z, Zhu Y, Yang F and Wu Z 2023 *Adv. Intell. Syst.* **5** 2200455
[7] Yan X, Li Z and Li C 2024 *Chin. Phys. B* **33** 028705
[8] Li Z, Xie W, Zeng J and Zeng Y 2023 *Chin. Phys. B* **32** 010503
[9] Lu J, Xie X, Lu Y, Wu Y, Li C and Ma M 2024 *Chin. Phys. B* **33** 048701
[10] Dai Y, Wang Z, Feng Z, Zou J, Guo W, Tan S, Yu R, Hu Y, Qian Z, Hu J, Xu Z, Zhu Y and Wu Z 2024 *Nanotechnology* **35** 475201
[11] Zou J, Zhu Y, Feng Z, Li X, Guo W, Qian Z, Qiu Y, Chen H, Xu Z, Dai Y and Wu Z 2024 *IEEE Trans. Circuits Syst. II Express Briefs* **71** 3288

[12] Zhang X, Wang X, Ge Z, Li Z, Wu M and Borah S S 2022 *Micromachines* **13** 2074
[13] Wang X, Bao X, Li X, Chen X, Liu G, Kang S and Iu H H 2024 *Nonlinear Dyn.* **112** 20353
[14] Yao P, Wu H, Gao B, Tang J, Zhang Q, Zhang W, Yang J and Qian H 2020 *Nature* **577** 641
[15] Lin P, Li C, Wang Z, Li Y, Jiang H, Song W, Rao M, Zhuo Y, Upadhyay N, Barnell M, Wu Q, Yang J and Xia Q 2020 *Nat. Electron.* **3** 225
[16] Yu A H, Tang M, Li G, Hou B P, Xuan Z W, Zhu B H and Chen T L 2021 *Atmosphere* **12**
[17] Gajera H K, Zaveri M A and Nayak D R 2022 *Int. J. Imaging Syst. Technol.* **32** 1774
[18] Li W, Shang Z, Qian S, Zhang B, Zhang J and Gao M 2022 *Expert Syst. Appl.* **205**
[19] Gao Z, Wang L and Wu G 2023 *Int. J. Comput. Vis.* **131** 363
[20] Babicz D, Kontar S, Peto M, Fulop A, Szabo G and Horvath A 2021 *IEEE Winter Conf. Appl. Comput. Vis. (WACV)*
[21] Sun M, Song Z, Jiang X, Pan J and Pang Y 2017 *Neurocomputing* **224** 96
[22] Opris I and Casanova M F 2014 *Brain* **137** 1863
[23] Wang Y and Sun Q Q 2021 *Cell Rep.* **36** 109733
[24] Sabour S, Frosst N and Hinton G E 2017 *Adv. Neural Inf. Process. Syst.* **30**
[25] Feng Z, Wu Z, Zou J, Cheng L, Zhao X, Zhang X, Lu J, Wang C, Wang Y, Wang H, Guo W, Qian Z, Zhu Y, Xu Z, Dai Y and Liu Q 2025 *Nat. Commun.* **16** 4925
[26] Zhang G, Kato J, Wang Y and Mase K 2015 *3rd IAPR Asian Conf. Pattern Recognit. (ACPR)* pp. 479
[27] Nguyen H N and Lee C 2018 *J. IEEE* **22** 14
[28] Ahmed K, Baig M H and Torresani L 2016 *14th Eur. Conf. Comput. Vis. (ECCV)* pp. 516
[29] Von Ahn L, Blum M, Hopper N J and Langford J 2003 *Adv. Cryptol. EUROCRYPT* **2003** 294
[30] Idros N, Rosli A, Aziz Z, Rajendran J and Marzuki A 2021 *Microelectron. Int.* **38** 46