

A progressive surrogate gradient learning for memristivespiking neural network

Shu Wang(王姝), Tao Chen(陈涛), Yu Gong(龚钰), Fan Sun(孙帆), Si-Yuan Shen(申思远), Shu-Kai Duan(段书凯), and Li-Dan Wang(王丽丹)

Citation: Chin. Phys. B, 2023, 32 (**6**): 068704. DOI: 10.1088/1674-1056/acb9f6
Journal homepage: http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

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

Simulation of the physical process of neural electromagnetic signal generation based on a simple but functional bionic Na^+ channel

Fan Wang(王帆), Jingjing Xu(徐晶晶), Yanbin Ge(葛彦斌), Shengyong Xu(许胜勇),Yanjun Fu(付琰军), Caiyu Shi(石蔡语), and Jianming Xue(薛建明)

Chin. Phys. B, 2022, 31 (**6**): 068701. DOI: 10.1088/1674-1056/ac3cac

Long range electromagnetic field nature of nerve signal propagation in myelinated axons

Qing-Wei Zhai(翟卿伟), Kelvin J A Ooi(黄健安), Sheng-Yong Xu(许胜勇), and C K Ong(翁宗经)

Chin. Phys. B, 2022, 31 (**3**): 038701. DOI: 10.1088/1674-1056/ac1e1a

Computational model investigating the effect of magnetic field on neural-astrocyte microcircuit

Li-Cong Li(李利聪), Jin Zhou(周瑾), Hong-Ji Sun(孙洪吉), Peng Xiong(熊鹏), Hong-Rui Wang(王洪瑞), Xiu-Ling Liu(刘秀玲), and Chang-Yong Wang(王常勇)

Chin. Phys. B, 2021, 30 (**6**): 068702. DOI: 10.1088/1674-1056/abea81

An electromagnetic view of relay time in propagation of neural signals

Jing-Jing Xu(徐晶晶), San-Jin Xu(徐三津), Fan Wang(王帆), and Sheng-Yong Xu(许胜勇)

Chin. Phys. B, 2021, 30 (**2**): 028701. DOI: 10.1088/1674-1056/abc0d2

Astrocyte and ions metabolism during epileptogenesis: A review for modeling studies

Meng-Meng Du(独盟盟), Jia-Jia Li(李佳佳), Zhi-Xuan Yuan(袁治轩), Yong-Chen Fan(范永晨), Ying Wu(吴莹)

Chin. Phys. B, 2020, 29 (**3**): 038701. DOI: 10.1088/1674-1056/ab6961

A progressive surrogate gradient learning for memristive spiking neural network

Shu Wang(王姝), Tao Chen(陈涛), Yu Gong(龚钰), Fan Sun(孙帆),
Si-Yuan Shen(申思远), Shu-Kai Duan(段书凯)†, and Li-Dan Wang(王丽丹)

College of Artificial Intelligence, Southwest University, Chongqing 400715, *China*

(Received 13 November 2022; revised manuscript received 7 January 2023; accepted manuscript online 8 February 2023)

In recent years, spiking neural networks (SNNs) have received increasing attention of research in the field of artificial intelligence due to their high biological plausibility, low energy consumption, and abundant spatio-temporal information. However, the non-differential spike activity makes SNNs more difficult to train in supervised training. Most existing methods focusing on introducing an approximated derivative to replace it, while they are often based on static surrogate functions. In this paper, we propose a progressive surrogate gradient learning for backpropagation of SNNs, which is able to approximate the step function gradually and to reduce information loss. Furthermore, memristor cross arrays are used for speeding up calculation and reducing system energy consumption for their hardware advantage. The proposed algorithm is evaluated on both static and neuromorphic datasets using fully connected and convolutional network architecture, and the experimental results indicate that our approach has a high performance compared with previous research.

Keywords: spiking neural network, surrogate gradient, supervised learning, memristor cross array

PACS: 87.19.L–, 87.19.ll, 89.20.Ff **DOI:** 10.1088/1674-1056/acb9f6

1. Introduction

In today's research, as an important branch of artificial intelligence, neural networks have attracted increasing attention of research institutions and enterprises, and have made great progress in many ways, such as image recognition[1] and problem decision making. Spiking neural networks (SNNs),[2] as the third generation of artificial neural networks,[3,4] in-spired by biological plausibility, are considered as new generation neural networks that take a step further toward brain-inspired intelligence, which simulate real biological brain neurons and synapses[5] and use models that best fit biological neuronal mechanisms to perform computations.

SNNs using spiking signals of 0 and 1 for information transfer for the purpose of simulating the process by which the synapses of organisms receive cumulative information, and neuron's different spike emission intervals contain a wealth of temporal information. The above characteristics improve biological realism[6,7] and prove it to be an effective tool for processing complex spatio-temporal information.[3] In addition, SNNs are event-driven and are only activated when an input spike is received. SNNs have shown great potential in solving deep learning problems owing to the advantages of high computational efficiency, low latency and low energy consumption, and applying to some devices based on brain-inspired chip[8,9] and computing platform.[10]

Currently, SNN's main training methods can be grouped into three categories: (1) Indirect supervised learning, i.e., ANN to SNN, this method first trains a non-spiking ANN and then uses the firing rate of each spike neuron to approximate neuron activity and convert it into SNN which retains the same structure.[11–13] With some strategies, conversions with almost no accuracy loss can be achieved.[14] However, ANN to SNN is only applicable to static datasets, not to neuromorphic datasets since its limitations of rate coding.[15] (2) Unsupervised learning, which learns weight modification of synaptic plasticity, like Hebb learning[16,17] and spike timing dependent plasticity (STDP).[18–20] Yet performance of those algorithms is lower than artificial neural network (ANN) because they are based on local learning rules. (3) Direct supervised learning. One learning type is improved supervised learning based on biological synaptic plasticity, e.g., Kheradpisheh *et al.* used supervised STDP learning algorithm to train deep spiking convolution neural networks for target recognition.[21] Another learning type is based on spike sequence convolution, e.g., Mohemmed *et al.* proposed the SPAN algorithm, which convolves the spike signal with a specific kernel function and converts it to an analog signal, allowing the Widrow–Hoff rule to be applied directly on the converted signal to adjust the synaptic weights.[22] Moreover, gradient descent is a pretty common optimization method for supervised learning. Bohte *et al.* proposed the first supervised learning algorithm based on error backpropagation named SpikeProp, which applies the simplified spike response neuron model.[23] Subsequently Tempotron[24] adjusted synaptic weights by controlling membrane potential, ReSuMe[25] modified the synaptic weights by supervising the error of the expected spike sequence of the input neuron in the teacher signal and the actual output spike sequence. Lee *et al.* training deep SNNs by considering neuron's membrane potentials as differentiable

†Corresponding author. E-mail: duansk@swu.edu.cn

© 2023 Chinese Physical Society and IOP Publishing Ltd

signals.[26] Jin *et al.* proposed the HM2-BP algorithm, which interprets the error backpropagation in SNN as two processes: the postsynaptic potential change induced by the pulse sequence and the back-propagation of the loss function defined by the frequency encoding.[27] Wu *et al.* proposed a back-propagation algorithm STBP (spatio-temporal backpropagation, STBP) that takes into account the spatio and temporal domains in SNN.[28]

Our main research interests are focused on gradient descent algorithm, and he most important step is gradient acquisition, while the activation function of the SNNs model is non-differential, and its complex neuron dynamics makes it difficult to train. In the previous research, the common solution is to replace the spike function or its derivative with a continuous function similar to this one in spike-based backpropagation algorithm,[29] such as the rectangular function, polynomial function, sigmoid function[28] and so on. How-ever, the above work is often based on static surrogate functions, while there is a gap between the real gradient and surrogate gradient, and results in accuracy loss. In this work we try to propose a dynamic approximation of the function.

To speed up calculation, a hardware structure is introduced. The memristor is a kind of non-volatile memory, the concept of it was proposed by Chua in 1971.[30] With the advantages of small size, low power consumption, easy integration and memory calculation, memristors stand out from the many neural analog computing hardware devices as one of the most desirable solutions to solve the limitations of the von Neumann architecture and realize the integration of storage and computation. Furthermore, ion migration in memory blockers is similar to the internal mechanisms of bio-synapses, enabling the realization of LTP (long-term potentiation), LTD (long-term depression), SRDP (spike-rate dependent plasticity) and other bio-synaptic properties.[31] In 2010, Lu Wei's team[32] proposed that the conductance of memristor could be precisely tuned by manipulating the ion migration in the device, and firstly used the memristor device to emulate the synaptic weight regulation and spike-timing dependent plasticity (STDP).

It will have great potential to simulate synapses in neural networks using memristor: (i) Memristor has the ability to perform matrix multiplication and cumulative (MAC) operations in parallel as application of Kirchhoff's law and Ohm's law, which are the core of various neural networks.[33] The performance of neural network model can be greatly improved and system energy consumption can be reduced by parallel accelerated MAC computation. (ii) The conductivity of memristor is adjustable and nonvolatile, and can be used as synaptic storage weights for forward propagation and backward weight update of neural networks. Yao P *et al.* developed a neuromorphic network for grey-scale face classification using a bidirectional analog 1024-cell-1T1R RRAM array, demonstrating faster convergence, higher accuracy, and lower energy consumption.[34] After that, they realized fully hardware-implemented memristor convolutional neural networks for the recognition of handwritten digits by integrating multiple memristor arrays.[35]

In this paper, the main contributions of this paper are as follows:

(i) We propose a progressive surrogate gradient function to solve the non-differential problem of SNNs fire activity. It can gradually approximate the fire function as the training epochs increase, with lower information loss and higher accuracy.

(ii) Moreover, a hardware deployment scheme on memristor is proposed to speed up calculations, and a GAP layer is introduced in the last layer to reduce the number of memristors.

(iii) Our model validation is on both static datasets (MNIST and Fashion-MNIST) and dynamic datasets (CIFAR10-DVS and DVS128-gesture), and it achieves state-of-the-art accuracy for nearly all tested datasets.

2. Related work

We take $\partial S/\partial u_i^{t,n}$ as the partial derivative of activation function with respect to membrane potential u, where $S(U(t)) = H(U(t)-\theta)$. Its derivative is 0 at each value except $U(t) = \theta$, where it cannot be defined, raising more problem in supervised training of SNNs.

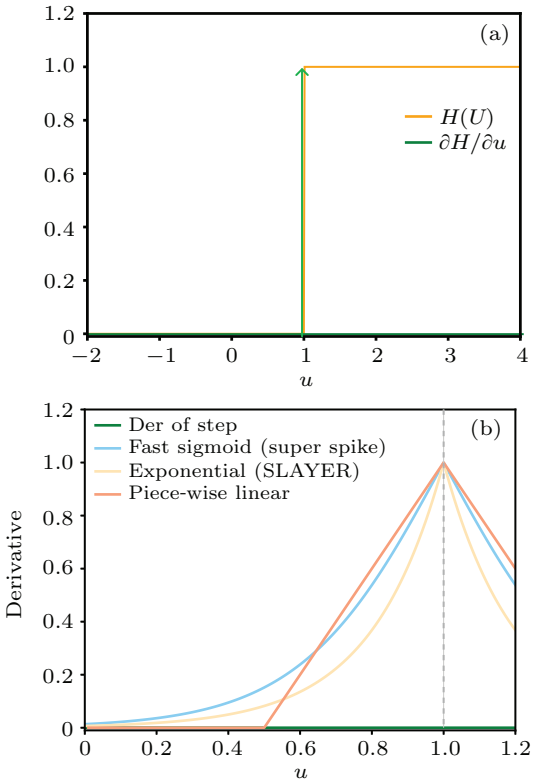


Fig. 1. Commonly used surrogate derivatives. (a) Step function and its derivative. (b) Commonly used surrogate derivatives. Green: derivative of step function, it is 0 everywhere except $U(t) = \theta$, where it cannot be defined. Yellow: exponential. Orange: piece-wise linear. Blue: derivative of fast sigmoid. The axes have been rescaled for ease of presentation.

Attention to previous research has been paid to introducing different approximate derivatives as surrogate gradient of spike activity. In the S4NN model given by Kheradpisheh *et al.*,[36] neuron's firing time of neurons is assumed to be equivalent to the output of RELU neuron. Li *et al.*[37] introduced difference gradients to quantitatively analyze the training behavior of SNNs and used the parameter control surrogate gradient function. Sumit *et al.* used an exponentially decaying function in SLAYER.[38] Friedemann *et al.* used fast sigmoid function as a suitable candidate in Super Spike,[39] and piecewise linearity is commonly chosen,[40–42] as shown in Fig. 1.

3. Surrogate gradient training framework

3.1. Leaky integrate-and-fire model in spiking neural networks

Spiking neurons are the basic unit of SNNs. Various simulation models of spiking neuron inspired by biology have been built for research, including the Hodgkin–Huxlecy model,[43] integrate-and-fire (IF) model, Izhikevich model,[44] leaky integrate-and-fire (LIF) model,[45] etc. We choose the common LIF neuron model (leaky integrate-and-fire model) for our work by considering the network complexity, the number of parameters and biological rationality.

The LIF model is described as follows: During a given time window T, membrane potential accumulates information from the pre-synapse until reaching threshold, then the neuron fires and membrane potential is reset to U_{reset}. Apart from that, the leakage term is introduced, which makes the membrane potential of neurons gradually decay to avoid the existence of previous memory after long-term interruption of neural activity. The dynamical evolution of the LIF model on a single spike neuron can be described in the following formula:

$$\tau_m \frac{\mathrm{d}v_i^l(t)}{\mathrm{d}t} = -(v_i^l(t) - v_{\text{rest}}) + I_i^l(t), \tag{1}$$

$$I_i^l(t) = \sum_i w_{ji}^l \sum_{t=1}^{T} S_i^{l-1}(t), \tag{2}$$

$$S_i^l(t) = H(v_i^l(t) - \theta_i^l), \tag{3}$$

where τ_m is the time constant that controls the change of membrane potential, $v_i^l(t)$ is the membrane potential of the i-th neuron of l-th layer at moment t, v_{rest} is the resting potential, $I_i^l(t)$ is the pre-synaptic input, $S_i^{l-1}(t)$ and w_{ji}^l are the spike sequence and synaptic weights of the i-th neuron in the l-th layer at the current moment, $H(x)$ is the Heaviside function, θ_i^l is the threshold.

However, it is difficult to train SNNs which are based on discrete data flow with Eqs. (1) and (2). They obey the following iterative updating rule (rewriting the differential equations as discrete difference equations):

$$\tau_m(v_i^l(t) - v_i^l(t-1)) = -(v_i^l(t-1) - v_{\text{rest}}) + I_i^l(t), \tag{4}$$

$$v_i^l(t) = v_i^l(t-1) + \frac{1}{\tau_m}(-(v_i^l(t-1) - v_{\text{rest}}) + I_i^l(t)). \tag{5}$$

Accordingly, Eq. (5) can be simplified to

$$v_i^l(t) = v_i^l(t-1) \cdot f(S_i^l(t-1)) + I_i^l(t). \tag{6}$$

3.2. Forward and backward propagation

The binary data streams of SNNs flow layer by layer through the network in the spatial domain (SD). Also, self-feedback is passed in the temporal domain (TD), as shown in Fig. 2.

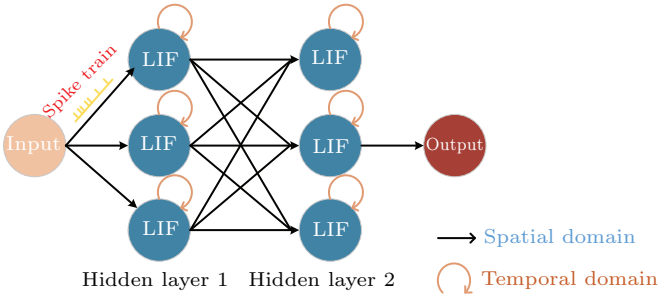


Fig. 2. Spatial and temporal dataflow in SNNs.

Referring to the STBP training framework and considering the transmission of errors over SD and TD.[28] In SD, layer-by-layer error backpropagation just likes the BP propagation for a general ANN, which means that each neuron accumulates the signals error with weights from the back layer. In TD, the neuron states are iteratively unfolded in the timing dimension with chain rules. The objective function is defined by using a least-squares loss in both SD and TD:

$$L = \frac{1}{2K}\sum_{k=1}^{K}\left\|\hat{y}_s - \frac{1}{T}\sum_{t=1}^{T} y_s^{t,N}\right\|_2^2, \tag{7}$$

where $\hat{y}_s$ is target vector, and $y_s^{t,N}$ is output vector of the last layer. In backpropagation, we obtain the partial derivative of w by chain rules as follows:

$$\begin{aligned}\frac{\partial L}{\partial w} &= \sum_{t=1}^{T}\frac{\partial L}{\partial v^{t,n}}\frac{\partial v^{t,n}}{\partial w^n} = \sum_{t=1}^{T}\frac{\partial L}{\partial v^{t,n}}\frac{\partial v^{t,n}}{\partial I^{t,n}}\frac{\partial I^{t,n}}{\partial w^n}\\ &= \sum_{t=1}^{T}\frac{\partial L}{\partial v^{t,n}} S^{t,n-1},\end{aligned} \tag{8}$$

$$\begin{aligned}\frac{\partial L}{\partial v^{t,n}} &= \frac{\partial L}{\partial S_i^{t,n}}\frac{\partial S_i^{t,n}}{\partial v_i^{t,n}} + \frac{\partial L}{\partial S_i^{t+1,n}}\frac{\partial S_i^{t+1,n}}{\partial v_i^{t,n}}\\ &= \frac{\partial L}{\partial S_i^{t,n}}\frac{\partial H}{\partial v_i^{t,n}} + \frac{\partial L}{\partial S_i^{t+1,n}}\frac{\partial H}{\partial v_i^{t+1,n}} f(S_i^{t,n}).\end{aligned} \tag{9}$$

3.3. Progressive surrogate gradient

It can be seen that there is a gap between the surrogate gradient and actual gradient, which leads to information loss and final accuracy reduction. To this end, we propose an progressive gradient substitution learning algorithm of SNNs by considering the update speed of the network and the accuracy of the gradient.

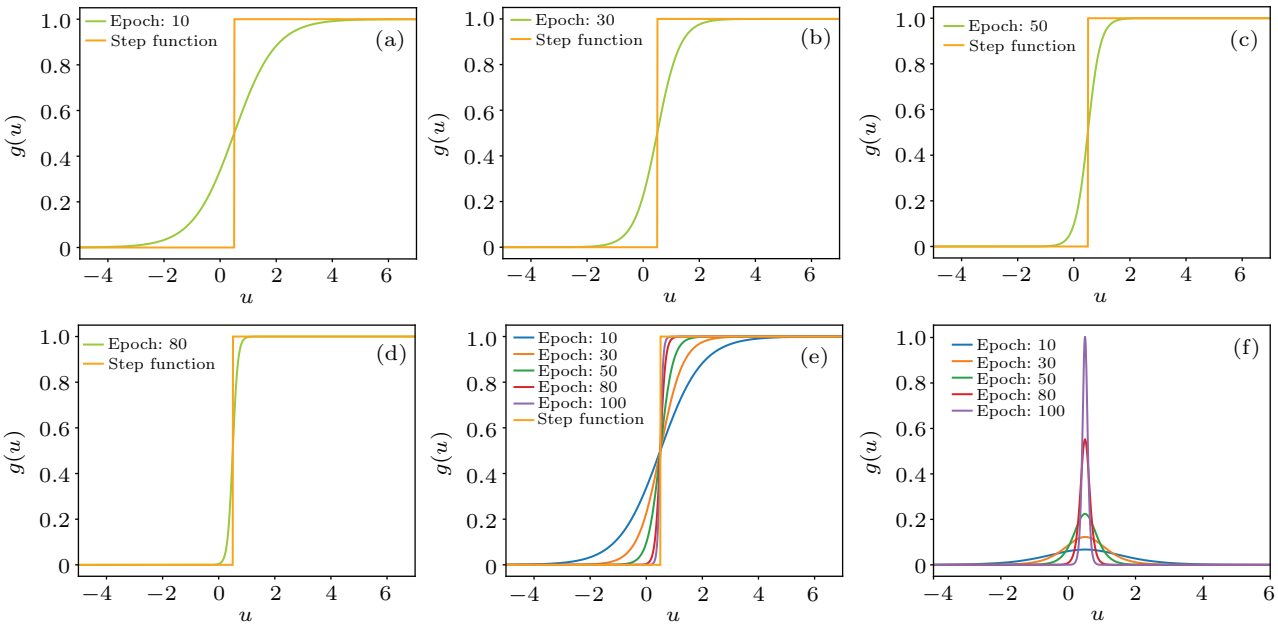


Fig. 3. Progressive surrogate function and its gradient at each stage (epoch = 100). (a)–(e) Shapes of the progressive surrogate function when epoch = 10, 30, 50, 80 and 100. (f) Progressive surrogate gradient in each epoch.

We introduce two dynamic parameters and define the function

$$g(x) = \frac{k}{1 + \mathrm{e}^{-\alpha(x-1)}}, \tag{10}$$

where $g(x)$ is the approximate substitution function of step function k, α is tunable parameters during training process and the value varies with the increase of training epoch. Here, k and t are defined as

$$\alpha = \mathrm{e}^{\frac{P\cdot\text{epoch}}{\text{Num_epoch}}}, \tag{11}$$

$$k = \max\left(\frac{1}{\alpha}, 1\right), \tag{12}$$

where P can be adjusted to control the shape of the function, epoch and Num_epoch are the current epoch and max epoch.

According to the derivation, the gradient of the substitution function at each place is

$$g'(x) = k \cdot \alpha \cdot S(x) \cdot (1 - S(x)), \tag{13}$$

$$S(x) = \frac{1}{1 + \mathrm{e}^{-\alpha(x-1)}}. \tag{14}$$

The approximate function we proposed and its gradient for each epoch stage are shown in Fig. 3 (maximum epoch set to 100 and hyper parameter P set to 2). There are two stages in the training process: (i) In the early stages of training, our approximate function generates gradient information on a larger range of voltage inputs during backpropagation than the derivatives of the original activation function, thus encouraging more spiking neurons to engage in learning and reducing the likelihood of falling into local optimal values. (ii) As the epoch increases, the function becomes gradually similar to the original activation function, which means that the surrogate gradient information is more accurate. At the maximum epoch, the approximation function is almost the same as the original activation function.

The fitting results of the function can be influenced by the parameters P, we will discuss the effect of its value on experimental results in Section 4. Therefrom, we consider that the gradient information $\partial H/\partial u_i^{t,n}$ can be approximated as $\partial H/\partial u_i^{t,n} \approx g'(u)$ in the training process.

4. Memristor and memristive unit

This section introduces the implementation scheme of the SNNs based on the memristor, the selected memristor model and the memristor neural network operation unit.

4.1. The VTEAM model

Since Hewlett Packard Labs achieved the first TiO_2 memristor device in 2008,[46] the research in the direction of memristors has continuously received increasing attention from scholars at home and abroad, and a variety of memristor models have been proposed one after another. Such as the Simmons tunnel barrier (STB) model presented by Abdalla and Pickett *et al.*,[47] which is precise but the mathematical model is too complex to establish a simulation experiment. Subsequently, the current threshold adaptive amnesia (TEAM) model was presented by Kvatinsky *et al.*,[48] including the TEAM model and the VTEAM model, with the difference whether the current or the voltage flowing through the memristor is used as the threshold.

Our work is carried out on a more widely used VTEAM model. Memory resistance changes when the voltage applied

to the device port exceeds a certain value. The VTEAM model can be expressed as

$$\frac{\mathrm{d}x}{\mathrm{d}t}=\begin{cases}k_{\mathrm{off}}\left(\frac{v(t)}{v_{\mathrm{off}}}-1\right)^{\alpha_{\mathrm{off}}}w_{\mathrm{off}}(x), & 0<v_{\mathrm{off}}<v,\\ 0, & v_{\mathrm{on}}<\mathrm{v}<v_{\mathrm{off}},\\ k_{\mathrm{on}}\left(\frac{v(t)}{v_{\mathrm{on}}}-1\right)^{\alpha_{\mathrm{on}}}w_{\mathrm{on}}(x), & 0<v_{\mathrm{on}}<v,\end{cases}\tag{15}$$

where x is an state variables denotes tunnel width, $v(t)$ is the voltage added to the device port of memristor, V_{on} and V_{off} are the thresholds of the model, k_{off}, k_{on}, α_{on}, and α_{off} are the fitted parameters, k_{off} represents being positive and k_{on} represents being negative, $w_{\mathrm{on}}(x)$ and $w_{\mathrm{off}}(x)$ are window functions used to control the value range of x.

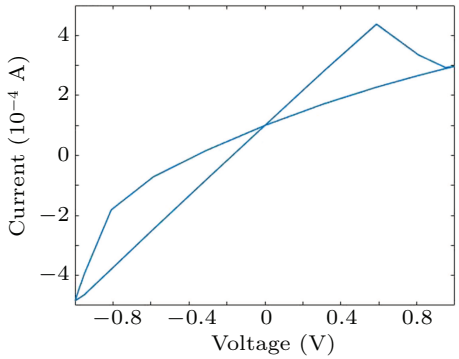


Fig. 4. Pinched loop of the VTEAM model.

Suppose that a linear relationship exists between the variable x and the resistance $R(x)$. The current–voltage relationship and the resistance of the memristor $M(x)$ in the VTEAM model are given as follows:

$$i(t)=\left[R_{\mathrm{on}}+\frac{R_{\mathrm{off}}-R_{\mathrm{on}}}{x_{\mathrm{off}}-x_{\mathrm{on}}}(x-x_{\mathrm{on}})\right]^{-1}\cdot v(t),\tag{16}$$

$$M(x)=\frac{v(t)}{i(t)}=R_{\mathrm{on}}+\frac{R_{\mathrm{off}}-R_{\mathrm{on}}}{x_{\mathrm{off}}-x_{\mathrm{on}}}(x-x_{\mathrm{on}}),\tag{17}$$

where $v(t)$ and $i(t)$ represent the current flowing through the memristor and voltage applied to memristor, respectively; x_{off} and x_{on} indicate the boundary value of the state variable x. R_{off} and R_{on} mean the maximal and minimal values of resistance, and λ is the fitted parameter.

A square wave signal of amplitude $\sin(2\pi t)$ is input to the memristor. The I–V curve is shown in Fig. 4, and it looks like a pinched loop. In experiment, when the applied voltage is below the threshold value, the state variable of the device remains approximately the same; when a voltage above a given threshold is applied, the memristor changes.

4.2. Memristive neural network

Typical memory network is realized by crossbar array, which is used to map the weight matrix of one layer of the model. The mapping relationship between a 3-by-3 memristor crossbar array and MAC operation is shown in Fig. 5(a), G_{ij} is conductance of memristor representing network weights, and voltage vector $[V_1, V_2, V_3]$ indicates the input vector of X. We can obtain the outputs of current that represents the output of the matrix by MAC operation $I_{\mathrm{output}}=\sum V_i G_{ij}$ using Kirchhoff's law and Ohm's law.

The peripheral circuit structure is shown in Figs. 5(b) and 5(c). The 1T1R structure is carried out, including three terminals connected to three different lines. One world line (WL) connected the gates from same row, one same bit line (BL) connected the device ports in the same column, and one source line (SL) connected the same signal sources in the same row.

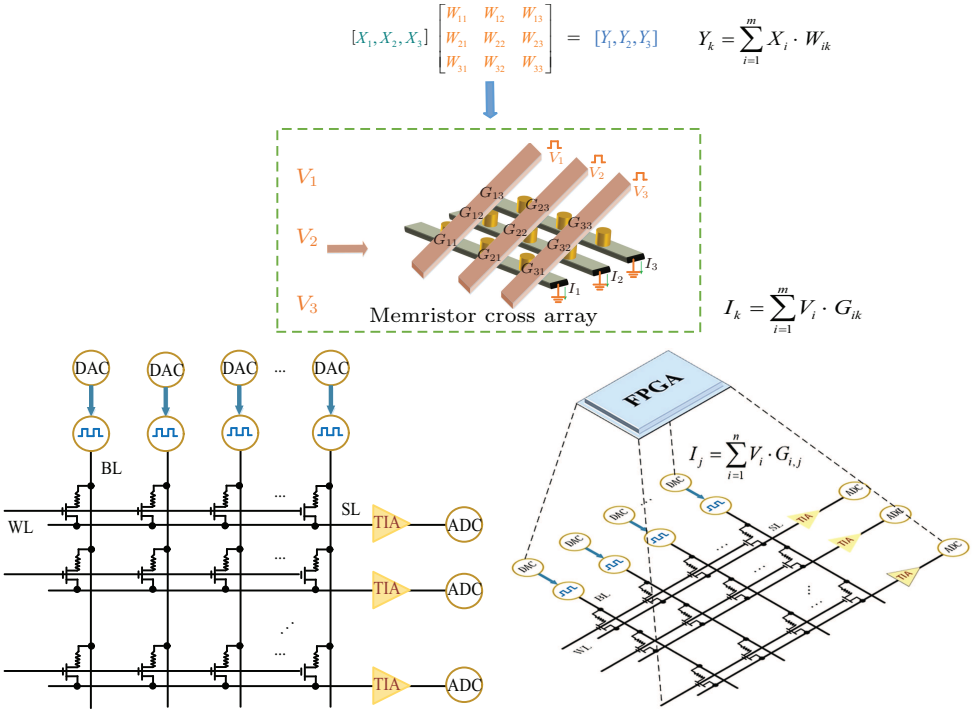


Fig. 5. Peripheral circuit structure of memristor cross array. (a) Memristor cross array, (b) mapping of matrix multiplication and addition operations (MAC) in memristor cross array, (c) computational schematic of memristive.

The data is converted into voltage pulses by the digital-to-analog converter DAC controlled by the peripheral FGPA, and input into in the memristor crossbar array. The memristor array is capable of performing parallel calculations, the results are converted to voltages in the form of currents through a trans-impedance amplifier (TIA) and captured by a high-speed analog-to-digital converter ADC.

SNNs are hardware friendly due to their storage method, event-driven nature. Compared to other neural network architectures, the SNN architecture-based implementation of the memristor array model enables lower power loss and higher computational efficiency.[40,50]

4.3. Memristive neural network operation unit

When a spiking convolution framework is used to process image recognition tasks, including convolutional (Conv) layers and fully connected layers. Among them, Conv layers are for feature extraction normally containing 2D Conv layers and pooling layers, and fully connected layers are for classification. Many previous works often used Poisson coding to convert the input image into a spike sequence input,[19,38,51] while this coding was considered to be degrading to performance of network,[20,52] and the suggested approach was to input the image directly and has the first convolutional layer for the coding task.

The structure of a general spiking convolution neural network (SCNN) is shown in Fig. 6. In the identification work of MNIST handwritten datasets, we use a specific network structure of 32C3-P2-64C3-P2-FC128-10. Here the C denotes the {Conv2D-Spiking Neurons}, and FC denotes {Fully Connected-Spiking Neurons}. In order to reduce scale of network parameters and the number of memristors used, we introduce the global average pooling layer (GAP) at the last layer of our network framework in place of a fully connected layer. Each layer of the memritive neural network deployment scheme is shown in Fig. 7. One memristor conductance can not directly represent the positive weight, thus a pair of memristors is required. Two memristor with positive conductance perform subtraction, it leads to two results: negative and positive.

For the input Conv layer, the $28 \times 28 \times 1$ image will be converted into $28 \times 28 \times L$ after convolutional computation using $3 \times 3 \times 1$ kernel with padding $= 1$ and stride $= 1$. Here, L denotes channels and its value is 32 in the first layer. To implement the convolution kernel unit on memristors, the 2D matrix is required to be flattened and scaled to a 1D column vector. The output of the Conv layer can be expressed as

$$Y_1 = \sum_{i=1}^{9} (X_i G_i^+ - X_i G_i^-). \tag{18}$$

There are average pooling layers after convolution computation, with size of 2×2. After the operation, an input image of $28 \times 28 \times 32$ will become $14 \times 14 \times 32$, and its channels would not be changed ($L = 32$ in the first layer). This process could be rewritten as a convolution operation as shown in Fig. 7(b), which is calculated by the following equations:

$$Y_1 = \frac{1}{4}(X_1 + X_2 + X_3 + X_4), \tag{19}$$

$$Y_j = \frac{1}{4}(X_{1,j} + X_{2,j} + X_{3,j} + X_{4,j}). \tag{20}$$

The last layer is the GAP layer, which is directly linked to the last convolution layer, and the global average pool of 2D images of each channel is given, resulting in an average value for each channel, as shown in the Fig. 7(c). Set the number of channels for the last layer of convolution to 10, which represents the number of categories in the dataset. After two convolution and pool-size operations as described above, the size of the image becomes $7 \times 7 \times 10$ and then a vector of 10 in length after passing through the GAP layer. The output Y_j can be written as

$$Y_j = \frac{1}{49} \sum_{i=1}^{49} X_{i,j}. \tag{21}$$

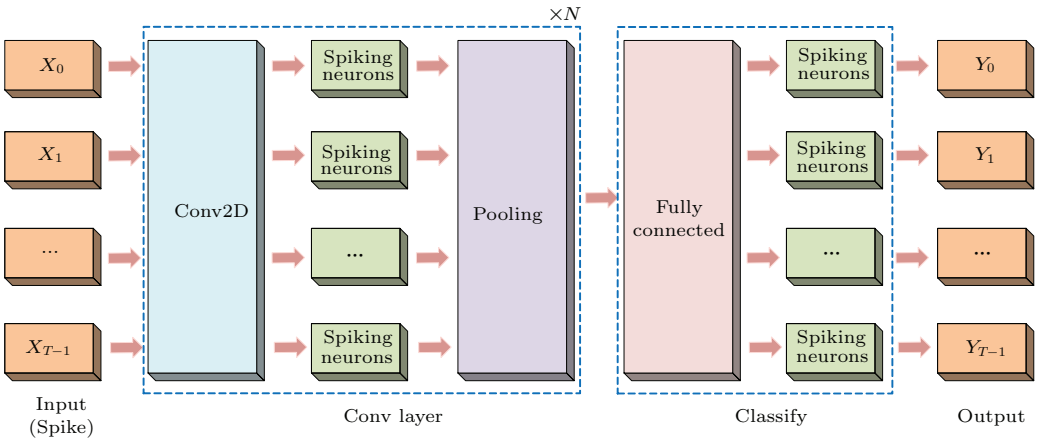


Fig. 6. Structure of a general spiking neural network.

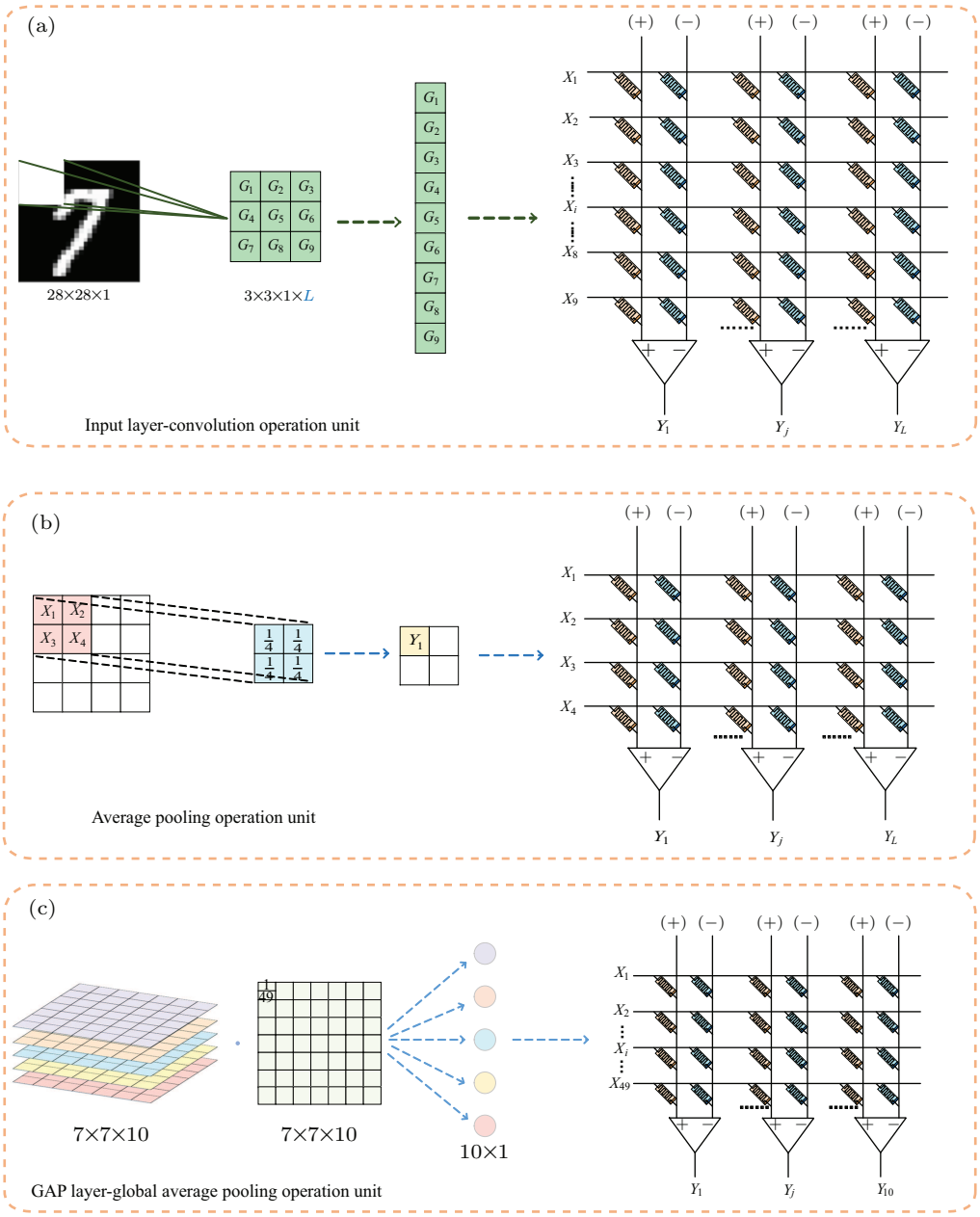


Fig. 7. Memristive operation unit of each layer.

5. Experimental results

Our experimental environment is based on the GPU of the Tesla P100. We deploy the algorithm to SNNs of convoluted and fully connected structures to test network accuracy on a number of datasets, including MNIST, Fashion MNIST, NMNIST, and DVS128.

5.1. MNIST

The MNIST dataset, organized by the National Institute of Standards and Technology (NIST), includes a set of 50000 images for training and a set of 10000 images for testing in a format of 28×28 grayscale images.

The results after training 150 epoch on the spike-MLP and spike-CNN structures are listed in Tables 1 and 2.

Table 1 compares the classification results with MLPs under similar structures, and Table 2 with those of CNN structures. It can be seen that our model works well compared to SNNs with the same structure, with up to 99.45% accuracy on MNIST datasets. In contrast, higher performance can be realized using our method.

We compare the impact of parameters P in Eq. (11). Figure 8 illustrates how different values of P influence the experimental results. It can be observed that the value of P that is too large causes the function to fit the original step function too early, resulting in no more neurons participating in subsequent training possibly falling into a local optimum; the value of P that is too small causes the curve of the alternative function to differ too much from the original function, reducing the final accuracy. As shown in Fig. 8(b), the red curve at epoch $= 20$ for $P = 10$ has early achieved a better fit, while the accuracy

curve corresponding to it can be observed in Fig. 8(a) with epoch = 20 or so as the cut-off point, and after reaching the highest accuracy rate, it has a decreasing trend into the local optimum.

Table 1. Comparison with typical MLP on MNIST.

Model	Model structure	Accuracy (%)
MLP (BP) (Non-spiking)	784-800-10	98.31
MLP (STBP).[28] (Spiking)	784-400-10	98.57
MLP (BP).[26] (Spiking)	784-800-10	98.71
MLP (This work) (Spiking)	784-800-10	98.86

Table 2. Comparison with Spiking CNNs on MNIST.

Model	Model structure	Accuracy (%)
SCNN (pre-training*)[41] (Spiking convolutional neuron network)	12C5-P2-64C5-P2-10	99.12
HM2BP[27]	15C5-P2-40C5-P2-300-1	99.42
SCNN(BP)[26]	20C5-P2-50C5-P2-200-10	99.31
This work	32C3-P2-32C3P2-128-10	99.45

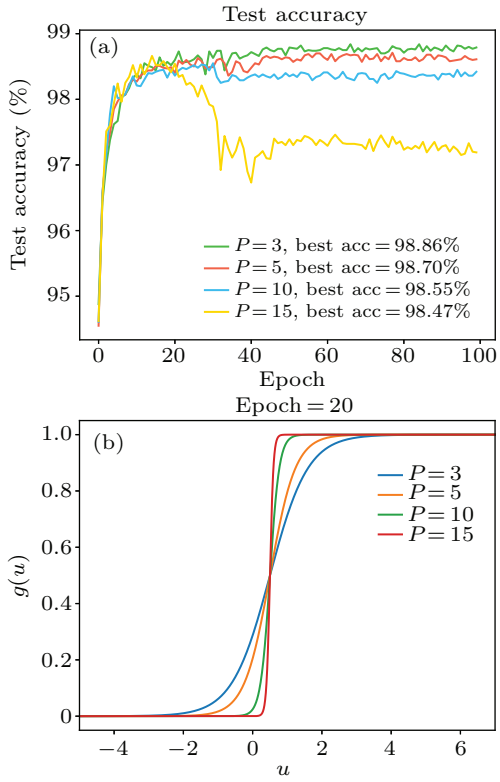


Fig. 8. Effect of different values P on the experimental results. (a) The results of different values of parameter P. (b) The experimental results under different values of parameters P when epoch = 20.

We conducted experiments on the MNIST dataset and compared them with the experimental results using rectangular functions, sigmoid function and other alternative functions. The performance of these alternative functions does not differ much. We get the best performance by adjusting the super parameters of the function, and the accuracy is between 98.42% and 98.51% using MLP network constructure. Our best accuracy (acc) is 98.86%, higher than all the functions here.

5.2. Fashion MNIST

The Fashion MNIST dataset from Zalando's research arm covers a total of 70000 different product images in 10 categories. It consists of a set of 60000 images for training and a set of 10000 images for testing. The rest of the format is identical to the MNIST dataset.

As shown in Table 3, our network accuracy is 92.34%, better than the same network framework non-spike CNN trained using the standard BP method.

Table 3. MNIST performance on Fashion MNIST.

Model	Model structure	Accuracy (%)
ANN[53]	400-400	89.01
HM2BP[27]	400-400	88.99
TSSL-BP[54]	400-400	89.80
This work	400-400	89.89
ANN[55]	32C5-P2-64C5-P2-1024-10	91.60
This work	32C3-P2-64C3 P2-1024-10	92.34

5.3. NMNIST

SNNs have a great advantage of processing time information, while dynamic datasets contain richer temporal information features. We evaluated our network performance by the N-MNIST dataset, which is a dynamic version of the dataset MNIST that is converted using scanning by dynamic vision sensor (DVS). The DVS collected events are triggered by pixel intensity changes. The events produce only when the pixel value changes by a user-adjustable threshold, and each event encodes pixel's spatial coordinates, reports a change and a time stamp indicating when the change occurs.

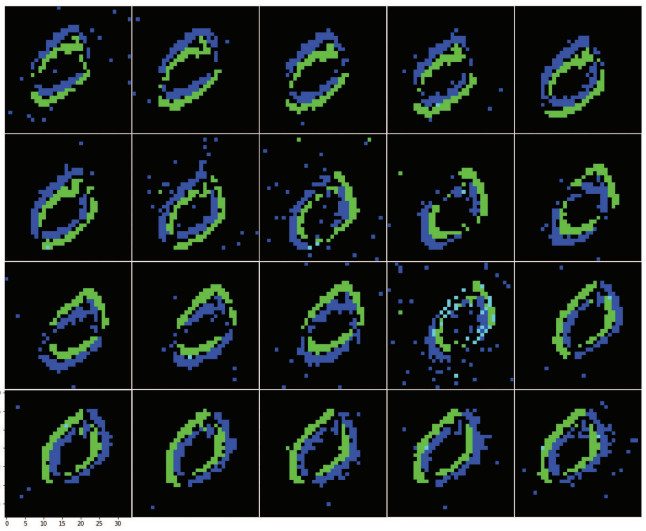


Fig. 9. Dynamic dataset of N-MNIST.

To facilitate training, we processed the data and converted the event data to frame data setting $T = 20$, as shown

in Fig. 9. Our network accuracy achieved 99.38% on the N-MNIST dataset, demonstrating the capabilities of the net to process temporal information on dynamic datasets.

Table 4. Performance on N-MNIST.

Model	Model structure	Accuracy (%)
CNN (BP)[12] (Non-spiking)	–	98.30
Spike-based BP[56]	–	98.28
Spike-based BP[28]	–	98.78
Spike-based BP[38]	–	99.20
This work	–	99.44

5.4. DVS128 gesture

We tested our algorithm on a DVS128 gesture dataset which includes 11 gesture categories for 29 subjects in 3 light conditions, taken by a DVS128 camera.[27] Our network works relatively well, with an accuracy rate of 97.57%, the most advanced work on SNN to date.

Table 5. Performance on DVS128 gesture.

Model	Model structure	Accuracy (%)
Spike-based BP[57]	–	92.01
SLAYER CNN[41]	–	93.64
Spike-based BP[58]	–	93.40
Spike-based BP[59]	–	95.54
This work	–	97.57

6. Conclusion and perspectives

In this work, we propose a progressive surrogate gradient learning method for gradient descent algorithm of SNNs to solve non-differentiable problem in training, a dynamic parameter influenced by epoch is introduced. The method can be used to reduce information loss and to improve network performance by dynamic approximation Heaviside function. Furthermore, we put forward a hardware implementation scheme to deploy the algorithm on memory resistor. Compared with other surrogate gradient functions, for both static and dynamic datasets our algorithm shows high accuracy and fast learning speed. It achieves state-of-the-art accuracy on all tested datasets, with test accuracy 99.45% on MNIST, 92.34% on Fashion MNIST, 99.44% on N-MNIST, and 97.57% on DVS128-gesture.

Acknowledgements

Project supported by the Natural Science Foundation of Chongqing (Grant No. cstc2021jcyj-msxmX0565), the Fundamental Research Funds for the Central Universities (Grant No. SWU021002), and the Graduate Research Innovation Project of Chongqing (Grant No. CYS22242).

References

[1] Vaila R, Chiasson J and Saxena V 2020 arXiv:2002.11843 [cs.NE]
[2] Gast R, Schmidt H and Knösche T R 2020 *Neural Comput.* **32** 1615
[3] Maass W, Natschläger T and Markram H 2002 *Neural Comput.* **14** 2531
[4] Maass W 1997 *Neural Networks* **10** 1659
[5] Hodgkin A L and Huxley A F 1952 *J. Physiol.* **117** 500
[6] Vreeken J 2003 *Spiking neural networks, an introduction*
[7] Cortex I T 2005 *Science* **310** 863
[8] Merolla P A, Arthur J V and Alvarez-Icaza R 2014 *Science* **345** 668
[9] Davies M, Srinivasa N and Lin T H 2018 *IEEE Micro* **38** 82
[10] Prezioso M, Merrikh-Bayat F and Hoskins B D 2015 *Nature* **52** 61
[11] Hunsberger E and Eliasmith C 2015 arXiv:1510.08829 [cs.LG]
[12] Neil D and Liu S C 2016 *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 22–25, 2016, Montreal, QC, Canada, pp. 2282–2285
[13] Rueckauer B, Lungu I A and Hu Y 2017 *Front. Neurosci.* **11** 682
[14] Deng S and Gu S 2021 arXiv:2103.00476 [cs.NE]
[15] Fang W, Yu Z and Chen Y 2021 arXiv:2007.05785 [cs.NE]
[16] Kelso S R, Ganong A H and Brown T H 1986 *Proc. Natl. Acad. Sci. USA* **83** 5326
[17] Hebb D O 2005 *The organization of behavior: A neuropsychological theory* (New York: Psychology Press)
[18] Querlioz D, Bichler Oa and Dollfus P 2013 *IEEE Trans. Nanotechnol.* **12** 288
[19] Diehl P U and Cook M 2015 *Front. Comput. Neurosci.* **9** 99
[20] Guo S, Yu Z and Deng F 2017 *IEEE Trans. Cybernet.* **49** 133
[21] Kheradpisheh S R, Ganjtabesh M and Masquelier T 2016 *Neurocomputing* **205** 382
[22] Mohemmed A, Schliebs S and Matsuda S 2012 *Int. J. Neural Syst.* **22** 1250012
[23] Bohte S M, Kok J N and La Poutré J A 2000 *SpikeProp: Backpropagation for networks of spiking neurons* (Belgium: The European Symposium on Artificial Neural Networks) **48** 419
[24] Gütig R and Sompolinsky H 2006 *Nat. Neurosci.* **9** 420
[25] Ponulak F and Kasiński A 2010 *Neural Comput.* **22** 467
[26] Lee J H, Delbruck T and Pfeiffer M 2016 *Front. Neurosci.* **10** 508
[27] Jin Y, Zhang W and Li P 2018 arXiv:1805.07866 [cs.NE]
[28] Wu Y, Deng L and Li G 2018 *Front. Neurosci.* **12** 331
[29] Neftci E O, Mostafa H and Zenke F 2019 *IEEE Signal Proc. Mag.* **36** 61
[30] Chua L. 1971 *IEEE Trans. Circuit Theory* **18** 507
[31] Li Y, Zhong Y and Zhang J 2014 *Sci. Rep.* **4** 4906
[32] Jo S H, Chang T and Ebong I 2010 *Nano Lett.* **10** 1297
[33] Yao P, Wu H and Gao B 2017 *Nat. Commun.* **8** 1
[34] Li J, Zhou G and Li Y 2022 *Artif Intell Rev* **55** 657
[35] Yao P, Wu H and Gao B 2020 *Nature* **577** 641
[36] Kheradpisheh S R and Masquelier T 2020 *Int. J. Neural Syst.* **30** 2050027
[37] Li Y, Guo Y and Zhang S 2021 *Adv. Neural Inf. Process Syst.* **34** 23426
[38] Vaezi M S and Vaezi A 2018 arXiv:1810.00864 [cond-mat.str-el]
[39] Zenke F and Ganguli S 2018 *Neural Comput.* **30** 1514
[40] Bellec G, Salaj D, Subramoney A *et al.* 2018 arXiv:1803.09574 [cs.NE]
[41] Esser S K, Merolla P A and Arthur J V 2016 *Proc. Natl. Acad. Sci. USA* **113** 11441
[42] Bohte S M 2011 *International Conference on Artificial Neural Networks* pp. 60–68. *Lecture Notes in Computer Science* Vol. 6791 (Berlin: Springer)
[43] Hodgkin A L and Huxley A F 1952 *J. Physiol.* **117** 500
[44] Izhikevich E M 2003 *IEEE Trans. Neural Netw.* **14** 1569
[45] Liu Y H and Wang X J 2001 *J. Comput. Neurosci.* **10** 25
[46] Strukov D B, Snider G S and Stewart D R 2008 *Nature* **453** 80
[47] Abdalla H and Pickett M D 2011 *IEEE International Symposium of Circuits and Systems (ISCAS)*, May 15–18, 2011, Rio de Janeiro, Brazil, pp. 1832–1835
[48] Kvatinsky S, Friedman E G and Kolodny A 2012 *IEEE Trans. Circuits Syst. I: Regular Papers* **60** 211
[49] Chen L, Gong C and Li C 2021 *Sustain Cities Soc.* **69** 102849
[50] Zhou W, Wen S and Liu Y 2023 *Neural Netw.* **158** 293
[51] Lee C, Sarwar S S and Panda P 2020 *Front. Neurosci.* **14** 119
[52] Rueckauer B, Lungu I A and Hu Y 2017 *Front. Neurosci.* **11** 682
[53] Zhang W and Li P 2019 arXiv:1908.06378 [cs.NE]
[54] Zhang W and Li P 2020 arXiv:2002.10085 [cs.NE]
[55] Xiao H, Rasul K and Vollgraf R 2017 arXiv:1708.07747 [cs.LG]
[56] He W, Wu Y J and Deng L 2020 *Neural Networks* **132** 108
[57] Xing Y, Di Caterina G and Soraghan J 2020 *Front. Neurosci.* **14** 1143
[58] He W, Wu Y J and Deng L 2020 *Neural Netw.* **132** 108
[59] Kaiser J, Mostafa H and Neftci E 2020 *Front. Neurosci.* **14** 424