SCIENCE CHINA
Information Sciences

• LETTER •

September 2019, Vol. 62 199103:1–199103:3
https://doi.org/10.1007/s11432-018-9607-8

Area-efficient memristor spiking neural networks and supervised learning method

Errui ZHOU, Liang FANG*, Rulin LIU & Zhensen TANG

Institute for Quantum Information & State Key Laboratory of High Performance Computing, College of Computer, National University of Defense Technology, Changsha 410073*, China*

Received 7 May 2018/Revised 16 August 2018/Accepted 13 September 2018/Published online 30 July 2019

Citation Zhou E R, Fang L, Liu R L, et al. Area-efficient memristor spiking neural networks and supervised learning method. Sci China Inf Sci, 2019, 62(9): 199103, https://doi.org/10.1007/s11432-018-9607-8

Dear editor,

Memristors have attracted a lot of attention since HP Labs first reported their memristive devices [1]. They have been employed in many fields, including non-volatile memory, image processing, and neuromorphic computing [2–4]. In neuromorphic computing, memristors are usually used as synapses because they can store synaptic weights by their conductance [4]. Most memristors have the structure of a crossbar array, which is area-efficient. With the cooperation of memristors and complementary metal oxide semiconductor (CMOS) neurons, it has become possible to implement low-power and area-efficient hardware implementation for neural networks [5].

Researchers consider spiking neural networks (SNNs) promising alternatives for conventional artificial neural networks (ANNs) in the future [6]. Many learning algorithms have been proposed to train SNNs. For instance, Bohte et al. [7] presented a method called SpikeProp that makes error backpropagation possible. This method approximates the threshold function as a continuous function in a small region with an appropriate learning rate. Moreover, spiking-time dependent plasticity (STDP) is a popular brain-inspired learning algorithm. Both supervised and unsupervised learning methods have been proposed based on STDP [8]. Zhang et al. [8] obtained an accuracy of 98.52% (the highest accuracy of spike-based SNNs) for MNIST by a STDP-like method.

Although SNNs can be simulated by software, this simulation involves multiple operations and parameters that require Von Neumann architecture with high computational capacity. If computations can be implemented through physical processes, fewer chip areas and less computational power will be required. Many efforts have been made to explore such a mode based on memristors. For example, Nishitani et al. [9] successfully trained SNNs based on three-terminal ferroelectric memristors by using a simple learning algorithm. However, complex spikes require complex hardware implementation, i.e., more hardware resources. To generate specific spikes, the hardware implementation of neurons requires integrators, comparators, spike generators, controllers, and so on. Moreover, there are many sub-connections between a pre-synaptic neuron and a post-synaptic neuron in SNNs [7, 9], which reduce the area efficiencies of crossbar arrays significantly during hardware implementation.

In order to simplify the design and increase the area efficiencies in hardware implementation, we propose practical memristor SNNs with greater area efficiency. Spikes in the memristor SNNs are simplified as step signals, which can directly be generated by comparators. Moreover, the number of synapses between a pre-synaptic neuron and a post-synaptic neuron is limited to one. However, the step function is not differentiable for the computation of the gradient descent. This means that SpikeProp-like supervised learning algorithms cannot be employed directly to train the proposed

* Corresponding author (email: lfang@nudt.edu.cn)

memristor SNNs. To overcome this issue, we approximate the step function as a modified sigmoid function, which is differentiable for computation. We then derive the gradient descent for the proposed memristor SNNs to implement training. Therefore, our work differs from previous studies in three key aspects: the spike function, the number of sub-connections, and the supervised learning method. To validate the proposed memristor SNNs and supervised learning method, the XOR problem, the Fisher Iris dataset and the MNIST dataset are employed.

Memristor SNNs. To describe positive and negative weights, we adopt the scheme where each synapse comprises two memristors, as shown in Figure 1(a). These two memristors are connected to the same output port of the pre-synaptic neuron but to different input ports of the post-synaptic neuron. One memristor is chosen to represent the positive weight, which equals the conductance in value. The other represents the negative weight, which equals the conductance in absolute value but has a negative sign.

Even though synapses are designed to have positive as well as negative values, implementation is performed by neurons. The structure of the neurons is also shown in Figure 1(a), and it has an integrator and a comparator. The integrator integrates currents flowing through the positive and negative memristors. We define the voltages integrated on the positive and the negative port as $V^+(t)$ and $V^-(t)$, respectively. When the voltage difference between $V^+(t)$ and $V^-(t)$ exceeds a predefined threshold (i.e., when the neuron is activated), a spike shaped step signal is generated by the neuron. Given that the clock frequency of present CMOS circuits can achieve several GHz, it is reasonable to think that neurons can generate step signals in the range of microseconds. In addition, there is no delay between the activation time and the spiking time.

The architecture of memristor SNNs is shown in Figure 1(b), and they comprise memristor crossbar arrays and neurons. To understand the proposed memristor SNNs in greater detail, Figure 1(b) is reduced to Figure 1(c), in which, there are input neurons, hidden neurons, and output neurons. There is only one synapse between each pre-synaptic neuron and post-synaptic neuron. The black blocks represent neurons, and the lines between neurons represent synapses. Note that there may be several hidden layers, which are not shown in Figure 1. When spikes (step signals) are generated by input neurons, they are propagated to hidden neurons via synapses. When the voltage difference of a hidden neuron exceeds the predefined threshold, the hidden neuron generates a spike and propagates it to neurons in the next layer. All inputs are binary in the proposed memristor SNNs; input neurons generate step signals at $t = 0$ ms when their inputs are 1 and generate nothing when their inputs are 0.

Detailed mathematical expressions of synapses and neurons are provided in Appendix A.

Supervised learning method. For convenience, the activation times of input, hidden, and output neurons are defined as t_i, t_h, and t_o, respectively. The weight of a synapse between the output layer and the hidden layer is represented as ω_{oh}, and the weight of a synapse between the hidden layer and the input layer is represented as ω_{hi}. We can define the error function as follows:

$$E = \frac{1}{2}\sum_{o\in O}(t_o - t_d)^2, \tag{1}$$

where t_d represents the desired activation times of the output neurons. The weight adjustment of the synapse between the output neuron o_m and the hidden neuron h_n is then computed by following equation:

$$\Delta\omega_{o_m h_n} = -\eta\frac{\partial E}{\partial\omega_{o_m h_n}}, \tag{2}$$

where η is the learning rate, which is positive, and $\partial E/\partial\omega_{o_m h_n}$ is the gradient. We follow the derivation process of SpikeProp. However, the step signal is not differentiable, which prevents the computation of the gradient. To compute the gradient, we approximate the step function as a modified sigmoid function, which is differentiable. The modified sigmoid function is expressed as follows:

$$f(t - t_i) = \frac{1}{1 + \mathrm{e}^{-s(t-t_i)}}, \tag{3}$$

where t_i is the step time, and s is used to modify the function shape. Therefore, $\Delta\omega_{o_m h_n}$ can be computed. Similarly, the weight adjustment of the synapse between the hidden neuron h_m and the input neuron i_n can be computed. The detailed derivation is described in Appendix B.

Applications. To validate the proposed memristor SNNs and supervised learning method, the XOR problem, the Fisher Iris dataset, and the MNIST dataset are employed. All simulations are implemented in MATLAB.

For the XOR problem, the numbers of neurons employed in different studies are approximately the same. However, the numbers of synapses vary greatly because the numbers of sub-connections vary. Our memristor SNN employs only 12 synapses, whereas Refs. [7, 9] used 320

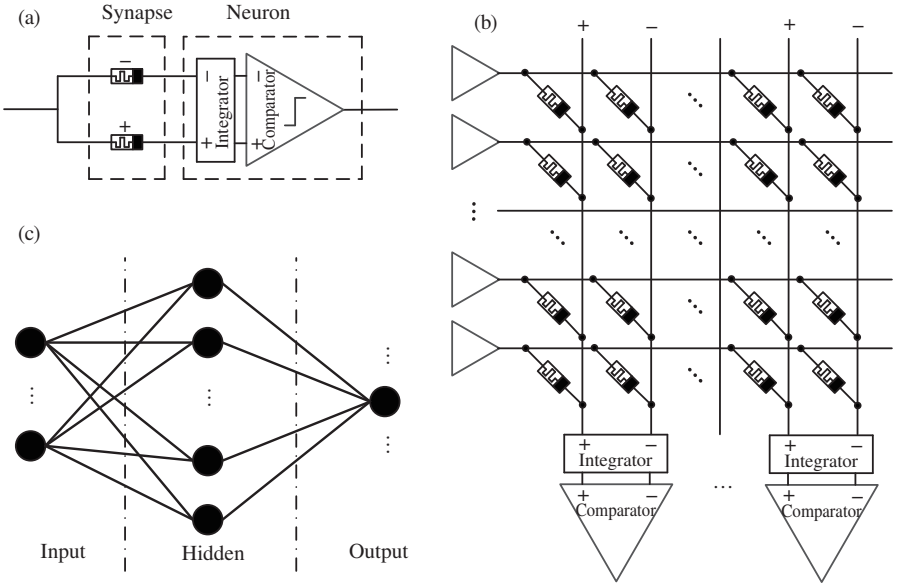

Figure 1 (a) Structures of the synapse and the neuron; (b) architecture of the memristor SNNs; and (c) schematic representation of the proposed memristor SNNs.

synapses. This shows that our memristor SNN uses 26 times fewer synapses, and our memristor SNN is more area-efficient.

In the case of the classification problem of the Fisher Iris dataset, we use several times fewer synapses in our memristor SNN than in [7], which means our memristor SNN is more area-efficient. Furthermore, we achieve an accuracy of 100% for the training set as well we the testing set with our memristor SNN, proving that our learning method obtains the highest accuracy compared with previous methods.

For the MNIST dataset, the accuracy of our study is 97.61%, slightly lower than 98.52% in [8]. Though the performance of our memristor SNN is not the highest, the significant advantage of our memristor SNN and the learning method is that they are designed for hardware implementation.

Detailed simulation results are described in Appendix C.

Conclusion. We propose an area-efficient memristor SNN for hardware implementation, the spikes of which are simplified as step signals. The number of synapses between a preneuron and a postneuron is limited to one. To train the memristor SNN, we present a supervised learning method that approximates the step function as a modified sigmoid function. Simulation results show that the memristor SNN is area efficient and the learning method can train the memristor SNN successfully.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 61332003, 61832007).

Supporting information Appendixes A–C. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Strukov D B, Snider G S, Stewart D R, et al. The missing memristor found. Nature, 2008, 453: 80–83
2 Zhu X, Tang Y H, Wu C Q, et al. Impact of multiplexed reading scheme on nanocrossbar memristor memory's scalability. Chin Phys B, 2014, 23: 028501
3 Zhou J, Yang X J, Wu J J, et al. A memristor-based architecture combining memory and image processing. Sci China Inf Sci, 2014, 57: 052111
4 Adam G C, Hoskins B D, Prezioso M, et al. 3-D memristor crossbars for analog and neuromorphic computing applications. IEEE Trans Electron Dev, 2017, 64: 312–318
5 Wu X, Saxena V, Zhu K, et al. A CMOS spiking neuron for brain-inspired neural networks with resistive synapses and in situ learning. IEEE Trans Circ Syst II, 2015, 62: 1088–1092
6 Maass W. Lower bounds for the computational power of networks of spiking neurons. Neural Comput, 1997, 8: 1–40
7 Bohte S M, Kok J N, La Poutré H. Error-backpropagation in temporally encoded networks of spiking neurons. Neurocomputing, 2002, 48: 17–37
8 Zhang T L, Zeng Y, Zhao D C, et al. A plasticity-centric approach to train the non-differential spiking neural networks. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI 2018), New Orleans, 2018. 620–627
9 Nishitani Y, Kaneko Y, Ueda M. Supervised learning using spike-timing-dependent plasticity of memristive synapses. IEEE Trans Neural Netw Learn Syst, 2015, 26: 2999–3008