Neural Comput & Applic (2018) 30:503–508
https://doi.org/10.1007/s00521-016-2700-2

ORIGINAL ARTICLE

Design of memristor-based image convolution calculation in convolutional neural network

Xiaofen Zeng[1,2] · Shiping Wen[1,2] · Zhigang Zeng[1,2] · Tingwen Huang[3]

Received: 22 July 2016 / Accepted: 3 November 2016 / Published online: 18 November 2016

Abstract In this paper, an architecture based on memristors is proposed to implement image convolution computation in convolutional neural networks. This architecture could extract different features of input images when using different convolutional kernels. Bipolar memristors with threshold are employed in this work, which vary their conductance values under different voltages. Various kernels are needed to extract information of input images, while different kernels contain different weights. The memristances of bipolar memristors with threshold are convenient to be varied and kept, which make them suitable to act as the weights of kernels. The performances of the design are verified by simulation results.

Keywords Memristor · Convolutional neural network · Image convolution computation

1 Introduction

Chua [1] was the first one to propose the hypothesis about the memristor. After studying about 40 years theoretically, HP Labs [2] made out the memristor physically in 2008, which attracts a lot of attention. Then various kinds of memristors emerged based on different materials. The memristance value can be varied under different voltage or current, and the value can be kept after withdrawing the voltage or current source applied on it, making memristors popular candidates for synapses. Besides nonvolatility, the memristor is characterized by nanoscale size and power efficiency, which enables them possible to be employed in neural networks [3–6], neuromorphic computing [7, 8], approximate computing [9] and memories [10]. Many researchers applied memristors in different architectures, such as learning architectures [3], computing architectures [11], Computation-in-Memory architectures [12], to get better performance. Moreover, different circuits based on memristors have been proposed. Adhikari et al. [3] presented the memristor bridge synapse used to implement random weight change algorithm in multilayer neural networks, while many scholars utilized memristor-based crossbar to realize matrix-vector operation [8, 13], neuromorphic character recognition [14], gradient-descent based learning algorithms [15, 16]. In addition, there are a lot of studies about memristors [17–21] in the literature.

Nowadays, the study of neural networks develops rapidly. Recently, Wen et al. [22] used neuroadaptive control approach to solve distributed consensus tracking problem for a class of multiagent systems with unmodeled dynamics and unknown disturbances. Many researches about them and their applications upsurge [23–26]. Convolutional neural network, as a kind of deep learning neural network was first inspired by the study of neural science, and a classical architecture of convolutional neural network was first proposed by Lecun et al. [27]. Compared with traditional neural networks, convolutional neural networks take advantages in weight sharing, which reduces the number of parameters need to be trained. In addition, it is good at recognizing images with displacement change, zoom, rotation and other forms of distortion. Therefore,

✉ Shiping Wen
wenshiping226@126.com

1 School of Automation, Huazhong University of Science and Technology, Wuhan, People's Republic of China

2 Key Laboratory of Image Processing and Intelligent Control of Education Ministry of China, Wuhan, People's Republic of China

3 Texas A & M University at Qatar, Doha 5825, Qatar

CNNs are very popular for pattern recognition and classification, such as human face recognition [28], traffic sign recognition [29], object recognition [30]. In advances in Neural Information Processing Systems, Krizhevsky et al. [31] used deep convolutional neural networks to classify more than a million images into 1000 different classes, which achieved a new state-of-the-art classification. Following the deep convolutional neural networks, Szegedy et al. [32] increased the depth and width of the network while keeping the computational budget constant to improve its ability of recognition and detection.

Quite a lot of reports about convolutional neural networks have been reported, and almost all of them are simulated via software. As we all know, software runs serially, but hardware computes inherently in parallel. However, the realization of the convolutional neural learning in fast, compact and reliable hardware is a difficult task. The critical problem is that hardware components cannot be utilized to store nonvolatile weight. In addition, convolution operations are complex for hardware to execute, containing too many multiplication operations and addition operations. Since memristors are nonvolatile with nanoscale size, it is necessary to apply them in convolutional neural networks to speed up the calculation. In this paper, the bipolar memristor with threshold is applied, which was put forward by Yuriy et al. [33]. This kind of memristor is nonvolatile, nanoscale and power efficient, with no differences to other kinds of memristors, but its memristance value is the same as previous value when the voltage is lower than the threshold.

Considering the traits of the convolution operation and the bipolar memristor with threshold, an architecture with memristors is designed to realize the convolution operation. The reminder of the article is structured as follows. Section 2 describes the memristor used in the design, as well as the modified convolution computation. Section 3 proposes the computation architecture and details the computation procedure and builds the computation circuits. Section 4 demonstrates the simulation results.

2 Background

2.1 Memristor model

Since researchers in HP Labs made the memristors, the interests in memristors upsurge. Many memristors based on different materials with diverse electrical properties have been discussed. Yuriy et al. [33] put forward the bipolar memristor with threshold. The memristor model is defined as the following:

$$I = x^{-1} V_{\mathrm{m}}, \tag{1}$$

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(V_{\mathrm{m}}) W(x, V_{\mathrm{m}}), \tag{2}$$

where x is the internal state variable, representing the memristance R. $f(x)$ is a function modeling the device threshold property, and $W(x)$ is a window function:

$$f(V_{\mathrm{m}}) = \beta(V_{\mathrm{m}} - 0.5(|V_{\mathrm{m}} + V_{\mathrm{t}}| - |V_{\mathrm{m}} - V_{\mathrm{t}}|)), \tag{3}$$

$$W(x, V_{\mathrm{m}}) = \theta(V_{\mathrm{m}})\theta(R_{\mathrm{off}} - x) + \theta(-V_{\mathrm{m}})\theta(x - R_{\mathrm{on}}), \tag{4}$$

where $\theta(x)$ is the step function, β is a positive parameter characterizing the rate of memristance change when $|V_{\mathrm{m}}| > V_{\mathrm{t}}$, and V_{t} is the threshold voltage. R_{on} and R_{off} are limiting values of the memristance R. In Eq. (4), the role of $\theta(x)$ is to restrict the memristance change to the interval between R_{on} and R_{off}. In order to avoid convergence problems, we modified the step function as:

$$\theta_{\mathrm{s}}(x) = \frac{1}{1 + \exp(-\frac{x}{b})}, \tag{5}$$

where b is a constant parameter. The absolute function can be adapted as:

$$\mathrm{abs}_{\mathrm{s}}(x) = x[\theta_{\mathrm{s}}(x) - \theta_{\mathrm{s}}(-x)]. \tag{6}$$

When a sinusoidal voltage source as shown in Fig. 1, is applied on the device, the change of state $x(t)$ is shown in Fig. 2.

By studying the pictures, we can conclude that the memristance is a constant value when the applied voltage is lower than the threshold, and its value varies between R_{on} and R_{off} under the opposite circumstance. Therefore, the threshold memristor model is adopted in the design and simulations.

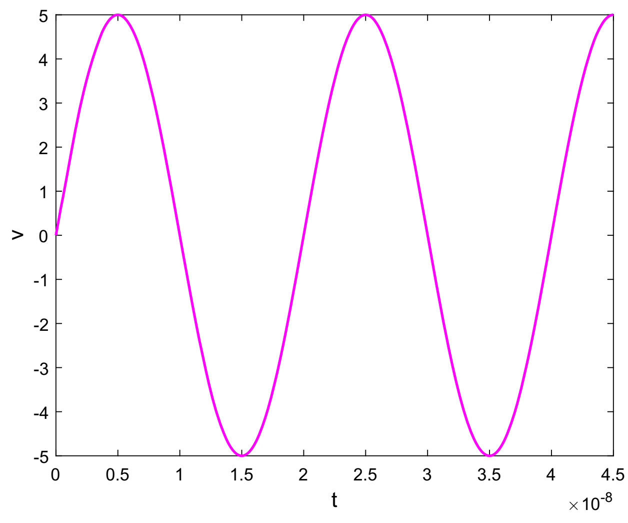

Fig. 1 The sinusoidal voltage source

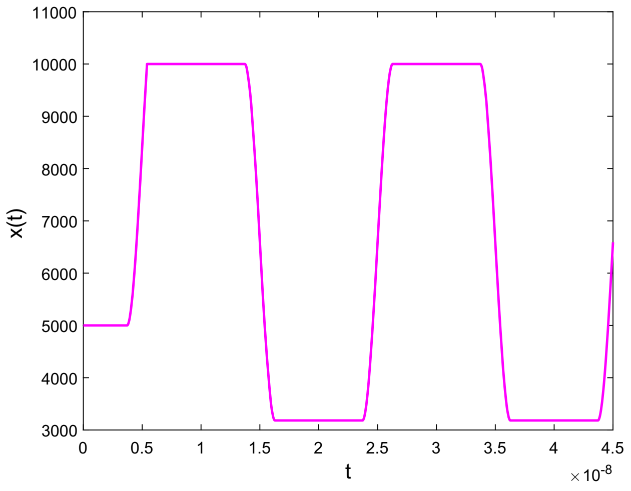

Fig. 2 The change of state variable $x(t)$

2.2 Image convolution computation

Image convolution computation aims to extract information from input images. Addison et al. [34] studied several kinds of neural networks' performance for feature extraction. Image convolution computation is an important part of CNNs. An image can be considered as a matrix, so the image convolution computation operation is the same as the matrix convolution computation operation. Set A is a $r_1 \times c_1$ matrix and B is a $r_2 \times c_2$ matrix. Generally, a 2-D convolution computation is defined as:

$$g(s,t) = \sum_{r=1}^{r_1+r_2-1} \sum_{c=1}^{c_1+c_2-1} f(r,c)h(s-r+1,t-c+1). \tag{7}$$

Following as Eq. (7) showed, it needs to perform $(r_1 + c_1 - 1) \times (r_2 + c_2 - 1)$ times multiplication and addition operations, which are very complicated. As we all know, it is time consumed to execute a multiplication operation by a software program. To complete a image convolution computation in convolution neural network efficiently, the algorithm is altered as:

$$Y_{11} = R\left(\sum_{i=1}^{M}\sum_{j=1}^{N} W_{ij}x_{ij}\right), \tag{8}$$

where Y_{11} is the first element of matrix Y, and the output after first convolution computation when the convolution kernel overlapped the input image. Integrated output is the size of $(I-M+1)\times(J-N+1)$. I, J represent the number of rows and columns of the input matrix A, respectively. W_{ij} is the weight of kernel. x_{ij} is the input converted from input image data or the subsampling layer, and R is a constant variable.

3 The computation architecture

3.1 Design of the architecture

As Eq. (1) shows, the current flowing through the memristor is the result of the multiplication of voltage and conductance ($G = x^{-1}$). So the multiplication operation can be conducted by the memristor. Kirchhoff's Current Law (KCL) describes that at any node (junction) in an electrical circuit, the sum of currents flowing into that node is equal to the sum of currents flowing out of that node. Based on KCL, a novel computation architecture for implementing Eq. (8) is proposed, as shown in Fig. 3.

Collecting currents from all branch circuits, the circle outputs the summation, which is multiplied by the resistance, then the product is the output. Now W_{ij} represents the conductance of a memristor, and R is a resistor, transforming the current to voltage for the subsampling conveniently. The architecture presented only can calculate one element of the output matrix.

In order to complete the whole computation, the calculation procedure is described by the Algorithm 1 in detail, where t is a temporary variable. The number of row and column of the kernel are M, N, respectively, while the number of row and column of the input image are I, J, respectively. Do as the Algorithm 1 shows once, you could

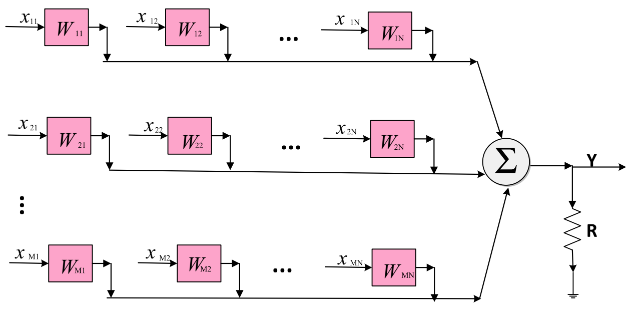

Fig. 3 The computation architecture

get a feature map. If you want to get different features, you need to make different kernels.

Algorithm 1 The Calculation Procedure

```
Initialization:
  set $j \Leftarrow 1$;
  for $j \leq J - N + 1$ do
    $i \Leftarrow 1$
    $k \Leftarrow j$
    for $i \leq I - M + 1$ do
      $h \Leftarrow i$
      $t \Leftarrow 0$
      $n \Leftarrow 1$
      for $n \leq N$ do
        $j \Leftarrow k + n - 1$
        $m \Leftarrow 1$
        for $m \leq M$ do
          $i \Leftarrow h + m - 1$
          $t = t + x(i,j) * w(m,n)$
          $m \Leftarrow m + 1$
        end for
        $n \Leftarrow n + 1$
      end for
      $Y(h,k) \Leftarrow t * R$
      $i \Leftarrow h + 1$
    end for
    $j \Leftarrow k + 1$
  end for
```

3.2 Building circuits

The circuit, using electrical elements to implement the function as Fig. 3 shows, is proposed in Fig. 4, where i_{11}, $i_{12}, \ldots, i_{mn}$ are currents flowing through different memristors. I represents a current-controlled current source (CCCS), whose value is the summation of $i_{11}, i_{12}, \ldots, i_{mn}$. Y_{11} is the voltage of the resistor, and it is a part of the feature map. For acquiring the whole one, two alternative methods can be adopted, the one being to copy the circuit to perform the function simultaneously, the other being to wait until the calculation completed. Obviously, the first method is more time saving. More circuit elements are required to build the circuit, in return. Therefore, a trade-off needs to be handled between speed and cost.

4 Simulation and analysis

HSPICE is compatible with most SPICE variations and takes advantages in convergence, accurate modeling and etc. Memristors are very small with nanometer size, and sensitive to the environment. In order to acquire the simulation results accurately, HSPICE is applied for simulating. When using the kernel

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix},$$

we could extract the edge information from the input images. The input image is shown in Fig. 5.

For the input image is a 5×5 matrix, and the kernel is a 3×3 matrix, performing the convolution operation as Algorithm 1 shows, there are 9 times convolution operations needed to be executed. In each convolution operation, the convolution kernel corresponding to the area is shown in Fig. 6. Figure 6a–g shows the area of the input image corresponding to the convolution kernel at each step. For example, at the first step, Fig. 6a is the input. After the input is convolved with the convolution kernel, the result is the first negative pulse as shown in Fig. 7. Similarly, the result of Fig. 6b convolved with the kernel is the second pulse in Fig. 7, and so on. After the calculation, the output simulated by HSPICE is shown in Fig. 7.

Apparently, there are four negative pulses in Fig. 7, whose number is the same as the black squares' in Fig. 5. Taking into account the characteristics of the image

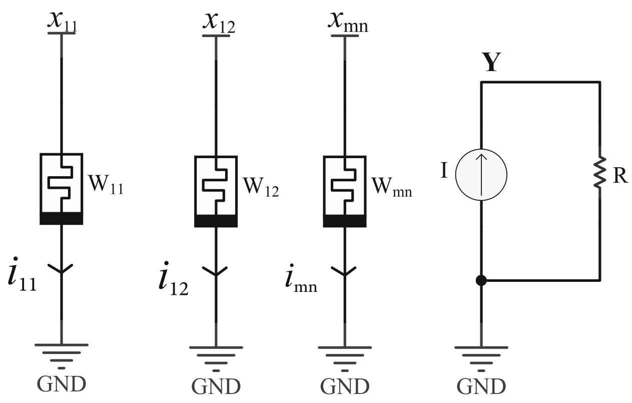

Fig. 4 Circuits to implement convolution operation

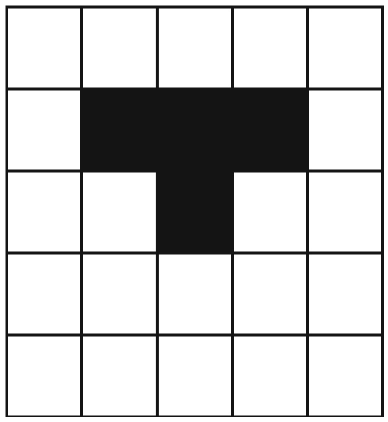

Fig. 5 Binary image used for testing

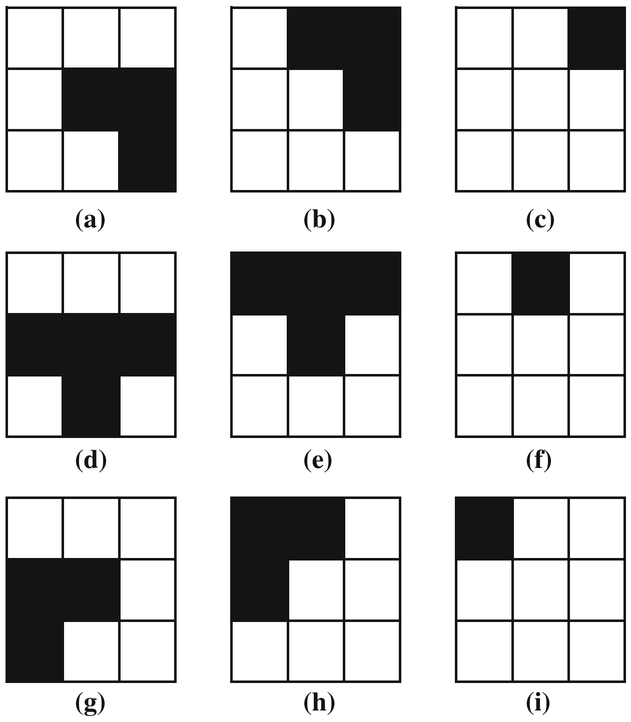

Fig. 6 Convolution kernel corresponding to the area of the input images in each convolution operation. **a** is the input at the first step, **b** is the input at the second step, **c** is the input at the third step, **d** is the input at the fourth step, **e** is the input at the fifth step, **f** is the input at the sixth step, **g** is the input at the seventh step, **h** is the input at the eighth step and **i** is the input at the ninth step

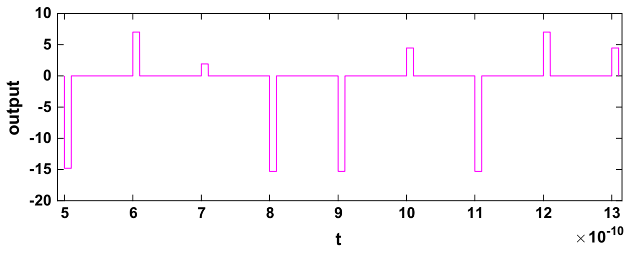

Fig. 7 Output after convoluting shown by MATLAB

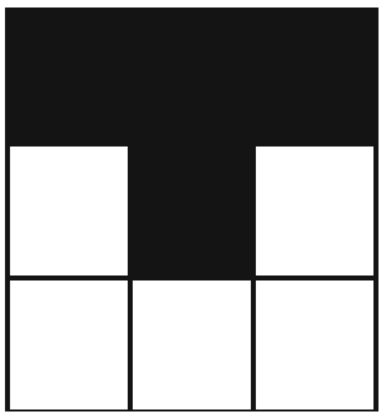

Fig. 8 Output after convoluting shown by a binary picture

convolution operation, we know that the feature map is a 3×3 matrix. So the white pixels of the feature map are the remaining of the input image, whose number is the same as the positive pulses. If the negative pulses are taken as the black pixels and the positive pulses as the black pixels, we could draw a picture as shown in Fig. 8. The simulation results verify proposed design eventually

5 Conclusion

Recently, CNNs take important roles in the field of computer vision, artificial intelligence and other areas. Developing convolutional neural networks in software cannot meet the commands of speed in today, so it is urgent to develop the method to achieve the hardware implementation. The effectiveness of the design is verified by simulations through HSPICE. In the future work, it is necessary to optimize the architecture and realize different kind of image processing.

References

1. Chua LO (1971) Memristor-the missing circuit element. IEEE Trans Circuit Theory 18(5):507–519
2. Strukov DB, Snider GS, Stewartand DR, Williams RS (2008) The missing memristor found. Nature 534(7194):80–83
3. Adhikari SP, Kim H, Budhathoki R (2015) A circuit-based learning architecture for multilayer neural networks with memristor bridge synapses. IEEE Trans Circuits Syst I 62(1):215–223
4. Ebong IE, Mazumder P (2012) CMOS and memristor-based neural network design for position detection. Proc IEEE 100(6):2050–2060
5. Liu XY, Zeng ZG, Wen SP (2016) Implementation of memristive neural network with full-function Pavlov associative memory. IEEE Trans Circuits Syst I Regul Pap 63(9):1454–1463
6. Li B, Chen L, Li CD, Huang TW, He X, Li H, Chen YR (2014) STDP learning rule Based on memristor with STDP property. In: Proceedings of the international joint conference on neural networks
7. Shi L, Pei J, Deng N, Wang D, Deng L (2015) Development of a neuromorphic computing system. In: Proceedings of the IEDM,
8. Hu M, Strachan JP, Li Z, Grafals E, Davila N (2016) Dot-product engine for neuromorphic computing: programming 1T1M crossbar to accelerate matrix-vector multiplication. In: Proceedings of the 53rd annual design automation conference
9. Li B, Gu P, Shan Y, Wang Y, Chen YR, Yang HZ (2015) RRAM-based analog approximate computing. IEEE Trans Comput Aided Des Integr Circuits Syst 34(12):1905–1917
10. Ho Y, Huang GM, Li P (2011) Dynamical properties and design analysis for nonvolatile memristor memories. IEEE Trans Circuits Syst I 58(4):724–736
11. Knag P, Lu W, Zhang Z (2014) A native stochastic computing architecture enabled by memristors. IEEE Trans Nanotechnol 33(2):283–293
12. Hamdioui S, Xie L, Nguyen H, Taoui M (2015) Memristor based computation-in-memory architecture for data-intensive applications. In: Proceedings of the conference on design, automation and test in Europe

13. Li HH, Liu CC, Yan B, Yang CF, Song LH, Li Z, Chen YR (2015) Spiking-based matrix computation by leveraging memristor crossbar array. In: Proceedings of the IEEE symposium on computational intelligence for security and defense applications (CISDA)
14. Ahmad MS, Hyunsang H, Moongu J, Jeon M (2014) Neuromorphic character recognition system with two PCMO memristors as a synapse. IEEETrans Ind Electron 21(6):2933–2941
15. Nair MV, Dudek P (2015) Practical gradient-descent for memristive crossbars. In: Proceedings of the conference on memristive systems (MEMRISYS)
16. Alibart F, Zamanidoost E, Strukov DB (2013) Pattern classification by memristive crossbar circuits using ex situ and in situ training. Nat Commun 4(3):131–140
17. Abdel-Kader RF, Abuelenin SM (2015) Memristor model based on fuzzy window function. In: Proceedings of the fuzzy systems (FUZZ-IEEE)
18. Li B, Wang Y, Chen YR, Li HH, Yang HZ (2014) ICE: inline calibration for memristor crossbar-based computing engine. In: Proceedings of the conference on design, automation and test in Europe (DATE14)
19. Chen YR, Tian W, Li H, Wang XB, Zhu WZ (2010) PCMO device with high switching stability. IEEE Electron Device Lett 31(8):866–868
20. Lucia VG, Arturo B, Luigi F, Fortuna L (2015) Memristor-based adaptive coupling for consensus and synchronization. IEEE Trans Circuits Syst I 62(4):1175–1184
21. Wen SP, Zeng ZG, Chen MZQ, Huang TW (2016) Synchronization of switched neural networks with communication delays via the event-triggered method. IEEE Trans Neural Netw Learn Syst. doi:10.1109/TNNLS.2016.2580609
22. Wen GH, Yu WW, Li ZK, Yu XH, Cao JZ (2016) Neuro-adaptive consensus tracking of multiagent systems with a high-dimensional leader. IEEE Trans Cybern. doi:10.1109/TCYB.2016.2556002
23. Lu JQ, Ho D, Cao J, Kurths J (2011) Exponential synchronization of linearly coupled neural networks with impulsive disturbances. IEEE Trans Neural Netw 22(2):329–336
24. Wen SP, Zeng ZG, Huang TW, Meng QG, Yao W (2015) Lag synchronization of switched neural networks via neural activation function and applications in image encryption. IEEE Trans Neural Netw Learn Syst 26:1493–1502
25. Liu D, Li H, Wang D (2013) Neural-network-based zero-sum game for discrete-time nonlinear systems via iterative adaptive dynamic programming algorithm. Neurocomputing 110:92–100
26. Cheng L, Hou ZG, Tan M, Lin YZ, Zhang WJ (2013) Neural-network-based adaptive leader-following control for multiagent systems with uncertainties. Neurocomputing 110:92–100
27. Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
28. Lawrence S, Giles CL, Tsoi AC, Back AD (1997) Face recognition: a convolutional neural-network approach. IEEE Trans Neural Netw 8(1):98–113
29. Mian ML, King HL (2015) Malaysia traffic sign recognition with convolutional neural network. In: Proceedings of the DSP
30. Wang JH, Lu JJ, Chen WH, Wu XM (2015) Convolutional neural network for 3D object recognition based on RGB-D dataset. In: Proceedings of the ICLR
31. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Proceedings of the NIPS
32. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the CVPR
33. Yuriy VP, Dalibor B, Massimiliano DV (2013) Reliable SPICE simulations of memristors, memcapacitors and meminductors, p 2717. arXiv:1307
34. Addison J, Wermter S, MacIntyre J (1999) Effectiveness of feature extraction in neural network architectures for novelty detection. In: Ninth international conference on artificial neural networks (ICANN 99)