ELectronics EXpress

LETTER

# Quaternary synapses network for memristor-based spiking convolutional neural networks

**Sheng-Yang Sun[1a)], Jiwei Li[1], Zhiwei Li[1], Husheng Liu[1], Haijun Liu[1b)], and Qingjiang Li[1]**

**Abstract** This paper proposes a method that renders the weights of the neural network with quaternary synapses map into the only four-level memristance of memristive devices. We show this method is capable of operating with a negligible loss in classification accuracy when the memristors utilized can store at least four unique values. Compared with other state-of-the-art methods, the method presented can achieve 98.65% accuracy under the 0.60M parameters. Systematic error analysis shows that the network can still reach over 95% accuracy under the condition of 95% yield of memristor crossbar array, 100 µV op-amp offset voltage and 0.5% Single-Pole-Double-Throw switches noise.

**Keywords:** memristor, convolutional neural networks, quaternary synapses network, neuromorphic computing
**Classification:** Integrated circuits

## 1. Introduction

With the development of artificial intelligence, neuromorphic computing has become a hot topic, and the next stage of high performance computing will dramatically improve the data processing and machine learning [1, 2, 3, 4].

In the recent years, the research of intelligent applications moves towards Internet of Things (IoT) edge computing, more and more devices are beginning to be miniaturized and integrated [5, 6, 7]. Due to the area limitation of hardware scale in mobile devices, the application requirements for specific target recognition often fail to achieve the desired results, which often consume excessive system resources or a large amount of energy [8, 9, 10, 11]. It is significant advantages to offer an embedded neuromorphic processing systems, which has the ability to solve complex problems while consuming very little power and area.

Concerning the hardware implementation for specific target recognition applications on portable mobile platforms, we proposed a three-layer spiking convolutional neural networks (CNNs) with parallel architecture (SCNNs) [12]. Although, the convolutional neural networks are competent in tolerating the random effects, such as the device variations or circuit noise [13], they may significantly degrade the recognition accuracy rate [14, 15].

[1]College of Electronic Science, National University of Defense Technology, Changsha 410073, China
a) sunshengyang13@nudt.edu.cn
b) liuhaijun@nudt.edu.cn

**DOI: 10.1587/elex.16.20190004**
**Received January 8, 2019**
**Accepted February 4, 2019**
**Publicized February 15, 2019**
**Copyedited March 10, 2019**

Despite the multilevel memristor devices that have appeared [16, 17, 18, 19], a better preparation technique has to be required since it still cannot be used for large-scale applications. Alternatively, we propose a method that renders the weights of the SCNNs with quaternary synapses in the network map into the only four-level memristance of memristive devices, as inspired by the recent trend of network pruning and parameter compression in the deep learning community.

This work builds on our prior memristor-based neuromorphic architecture [12], our previous implementation requires memristor device with multiple levels. In this work our goal is to find four-level resistances values within the memristor device resistance range, even when the memristive device cannot support many states stably, the network can still achieve a high accuracy rate as much as possible.

The rest of this paper is organized as follows, Section 2 describes the basic structure of spiking convolutional neural networks and the quaternary synapses network. Section 3 exhibits the simulation performance of our proposed method. The final Section 4 concludes the paper.

## 2. Memristor-based spiking convolutional neural networks with quaternary synapses

### 2.1 SCNNs architecture

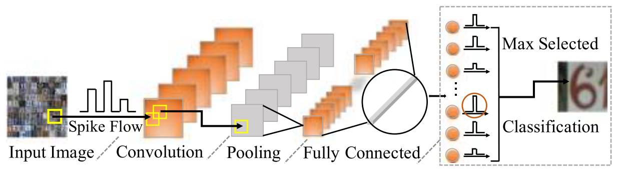

**Fig. 1.** The SCNNs consists of three modules, behind the input layer is convolution layer, followed by a Max Pooling layer and a fully connected layer. We use the absolute activation function to connect the convolution layer and the pooling layer, and we only use the bias at the last full connection layer.

The proposed architecture of SCNNs [12] is shown in Fig. 1. The input image transmitted to the network is represented in analogue method, whose information value is carried by the pluses with different amplitude. Compared with the other deep convolutional neural networks [20, 21, 22, 23], we only adopt one convolution layer and one pooling layer, which is to facilitate the hardware implementation for achieving full parallelism with lower consumption. At the end of the fully connected layer, the number of neurons is the same as the number of categories, and we choose the neuron with the largest value as its classification output. In the actual hardware implementation, the largest value is the maximum pulse amplitude.

## 2.2 Hardware implementation

As mentioned above, the SCNNs is a simplied CNNs architecture. After the network training is finished, the weight matrix also has some negative weights. Therefore, the converted mapping method is needed to be applied. We assume that $W$ represents a $2 \times 2$ weights matrix, and it includes positive and negative values. Then, the $W$ is converted to the $2 \times 4$ matrix so that the memristor crossbar can easily calculate the weighted-sum with the amplifiers. Each original value is extended in two parts, $W^+$ and $W^-$. If one element is a positive value, then the value is defined in $W^+$, and the value of $W^-$ is zero. In contrast, the negative element value is defined in $W^-$ and the $W^+$ is zero. Similarly, if the arrangement of the memristor in the array corresponds to the converted matrix, the $R_{off}$ takes the place of the zero element. Fig. 2 is a simple demo about performing a convolution computation in a memristor crossbar. A two-dimensional input image is converted into a one-dimensional electrical signal as an input, and the weighted-sum computation is completed by the memristive crossbar array.

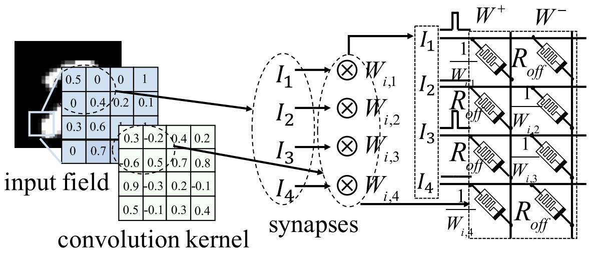

**Fig. 2.** The example illustrates the process of convolution by using the memristor crossbar. The $2 \times 2$ size of image is selected as a demo. The input is represented by the voltage pulse with amplitude information, and the weights are stored in the memristor crossbar.

The Abs and pooling circuits are shown in Fig. 3. The Abs activation function module follows the convolution layer. The Abs mainly consists of two op-amps and two diodes. The $V_{in}$ terminal receives the spike signals from the convolution layer. The $V_{out}$ terminal generates an activation spike and sends it to the pooling layer. The pooling module requires three op-amps and three Single-Pole-Double-Throw switches (SPDT). The $V_{in}^i$ receives the spike signal that has been activated, and the maximum voltage will be sent to the fully connected layer by $V_{out}$.

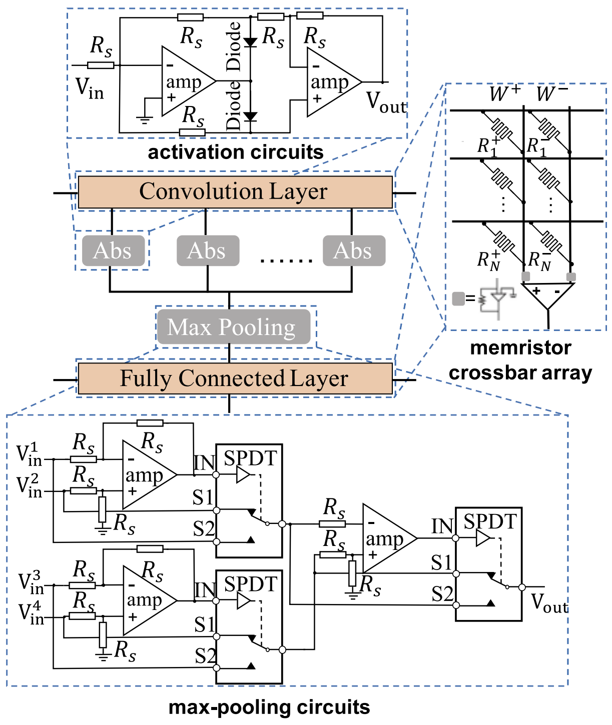

**Fig. 3.** Modules of convolutional layer, fully connected layer, Abs activation function and max pooling.

## 2.3 Quaternary synapses network

Considering the realistic characteristic of the $TaO_x$ memristor device model [24], the trained matrix weights should be converted to conductivity values that fall within the bounded range of a memristor crossbar.

$$S_i^C = S_i^F = \frac{S_{max} - S_{min}}{W_{max}} \cdot |W_i| + S_{min} \qquad (1)$$

Equation (1) shows how the synaptic weights are converted to conductivity values, and the conversion procedure is implemented on the software platform. $S_i^C$ and $S_i^F$ represent the conductance values at the convolutional and fully connected layers, respectively. The $S_{max}$ and $S_{min}$ indicate the maximum and minimum conductances, respectively. The $W$ is the original SCNNs weights set, and the $W_{max}$ represents the maximum absolute value of the weights set, and the $S_i$ is the conductance of the memristive array.

Our goal is to find four-level resistances values within the memristor device resistance range that we called the Quaternary Synapses Network (QSN). Even when the memristive device cannot support many states stably, the network can still achieve a high accuracy rate as much as possible.

**Algorithm 1** Quaternary Synapses Network Algorithm

```
Require: Mapped weights set S
Ensure: Four conductivity values $C_1, C_2, C_3, C_4$
  Initialize $\gamma$ and intervals $B$
  $C_4 \leftarrow \mathbf{O}$, $Cost_{min} \leftarrow \infty$
  For convenience, $\alpha \leftarrow (S_{max} - S_{min})/2^B$
  for $i = 1:1:2^B$ do
    for $j = i+1:1:2^B$ do
      for $k = j+1:1:2^B$ do
        put the $i, j, k$ in the set D
        $W_m = S_{min} + \alpha \cdot D_m$, which $D_m \in D$
        $sum \leftarrow \mathbf{O}$
        generate the F by W and $C_4$
        for $m = 1:1:length(\mathbf{S})$ do
          $Dist_{min} = min\{|S_m - F_\phi|\}$, which $F_\phi \in \mathbf{F}$
          $sum = sum + Dist_{min} \cdot \gamma$
        end for
        if $sum < Cost_{min}$ then
          $Cost_{min} = sum$
          $\{C_1, C_2, C_3\} = D$
        end if
      end for
    end for
  end for
  $C_i = S_{min} + \alpha \cdot C_i$, which $C_i \in \mathbf{C}$ and $i \neq 4$
  return $C_1, C_2, C_3, C_4$
```

The weights of the SCNNs are distributed in the convolution layer and the fully connected layer. First, we need to map the weights of the convolutional layer and the fully connected layer to the conductance values by using Eq. (1). Next, we need to determine the four conductivity values $C_1$, $C_2$, $C_3$, and $C_4$ (for the resistance of the memristors $R_1$, $R_2$, $R_3$, and $R_4$, respectively). To make it easy to represent negative weights in the memristive array, we fix a conductance of $C_4 = 0$ ($R_{off}$). Since two memristors combined in the array can correspond to 13 weights, we use $F$ to represent the set of weights consisting of the four-level memristor state (i.e., $F = \{C_1, C_2, C_3, C_4, C_2 - C_1, C_3 - C_2, \ldots\}$). We define the conversion error loss function $\varphi$ to determine $C_1$, $C_2$, and $C_3$.

$$Q_{ij} = \begin{cases} |S_i - F_j| & |S_i - F_j| = min\{|S_i - F_j|\} \\ 0 & other \end{cases} \tag{2}$$

$$\gamma = \begin{cases} 1/N_C & S_i \ in \ S^C \\ 1/N_F & S_i \ in \ S^F \end{cases} \tag{3}$$

$$\varphi = \sum_{i=1}^{|S|} \sum_{j=1}^{|F|} Q_{ij} \cdot \gamma \tag{4}$$

The definition $\varphi$ is described as Eq. (4), where $S$ is a set of mapped weights. Since there are different numbers of weights in the convolutional layer and fully connected layer, the weights of the convolutional layer and the fully connected layer are weighted with $\gamma$ when the conversion error is calculated. The value of $\gamma$ depends on the number of weights in the convolution or fully connected layer, which means that $\gamma = 1/N_C$ or $\gamma = 1/N_F$, where $N_C$ and $N_F$ are the numbers of weights of the convolutional or fully connected layer, respectively. When the conversion error loss function $\varphi$ reaches the minimum, $C_1$, $C_2$, $C_3$, and $C_4$ are the conductance values that we required (algorithm 1 describes the process). Fig. 5(a) shows the distribution of device conductance values in the QSN memristive array, and devices in the array will only use one of the four states of memristors.

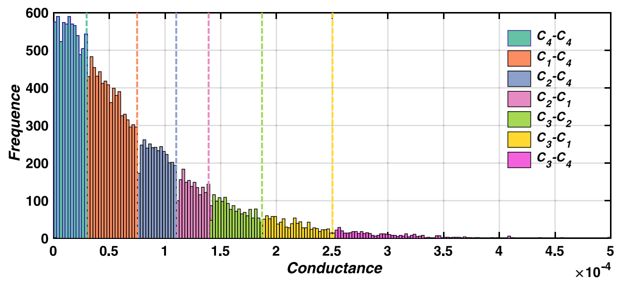

**Fig. 4.** Mapping the new conductance values by combining four states.

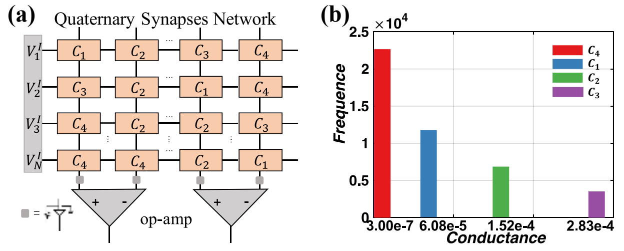

**Fig. 5.** (a) Quaternary Synapses Network in SCNNs. The device in the array will only use one of the four states of memristors. (b) Distribution of synapses conductance mapped to four values.

$$S_i' = \sum_{j=1}^{|F|} Q_{ij}, \quad i = 1, 2, \ldots, |S| \tag{5}$$

Equation (5) shows that we will choose a value that is closest to the original value as the new weight of the network, and $S_i'$ is the new conductance in the crossbar. Fig. 5(b) shows the distribution of the weights by using the four-state synapses network, and it can be seen that the high-resistance state accounts for the majority of memristors in the crossbar. In the actual implementation of the algorithm, we divide the abscissa into $2^B$ (e.g., $B = 8, 9, 10\ldots$) intervals to facilitate the statistics. That is, during the search process, we search the four states in the $2^B$ quantized values. Fig. 4 shows the distribution of the synaptic weights by a combination of the four-state resistances.

## 3. Simulation performance

### 3.1 Experimental settings

All experiments are conducted using an Intel Core i5 (2.5 GHz), 8 GB DDR3 and an Intel HD Graphics 400 graphics card. To reflect the practical application of algorithm performance better, we do not use any data augmentation technologies [25, 26, 27]. This work is implemented with the Theano python open-source library, and we also use LTspice for some circuit simulations. In the experiments, we chose the MNIST dataset for verification, we normalize the gray value of the image as the input voltage, that is, the input voltage range is 0–1 V. In this paper, we refer the data from a fabricated $TaO_x$ memristor device [24]. These devices can be repeatedly programmed to different target resistance states from 2 K to 3 MΩ and show the needed linearity at sufficiently low voltages ⩽0.3 V.

## 3.2 Experiments on datasets

The MNIST dataset [20] consists of 60,000 training examples and 10,000 test examples of handwritten digits. Each input image is a $28 \times 28$ grey scale image with a corresponding label $l \in [0, 9]$, indicates different digits. The size and simplicity of the MNIST dataset make it convenient and quick to test models on.

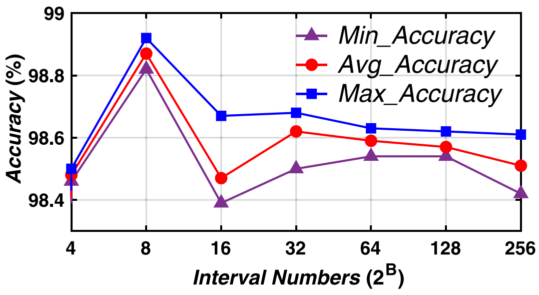

**Fig. 6.** Recognition accuracy with using the Quaternary Synapses Network.

Fig. 6 illustrates the relationship between the number of intervals and the recognition accuracy rate. We can find that the QSN achieves 98.4% recognition accuracy rate at least. The overall recognition rate is the highest when the interval number of 8 is set in the algorithm ($B = 3$), the average accuracy can reach 98.85% which is only 0.15% decay compared with the highest accuracy.

**Table I.** Performance comparison with other publications

| Method | Parameters | Memristor Amount | Accuracy |
|---|---|---|---|
| BinaryConnect [28] | 17.32 K | 1.07 M | 96.52% |
| BinaryNet [29] | 13.12 K | 0.80 M | 98.16% |
| Gated-XNOR [30] | 11.89 K | 0.73 M | 98.38% |
| **This work** | **9.73 K** | **0.60 M** | **98.65%** |

The performance achieved in this work was compared with other state-of-the-art designs (ensure accuracy ⩾95%). As listed in Table I, it can be seen that our proposed method offered comparable performances.

## 3.3 Systematic error analysis

As we mentioned in the previous section, there are some op-amps in the activation function and pooling module. The amplifier at each module may also impact recognition accuracy. To consider this effect, we assume that each op-amp in the circuit's voltage input offset error, associated gain error and voltage noise. The op-amps error model is constructed as follows:

$$V_{out} = V_{in} \cdot E_{gain} + E_{offset} + E_{noise} \quad (6)$$

where $E_{gain}$, $E_{offset}$, and $E_{noise}$ denote the op-amps voltage gain error, the offset voltage, and the voltage noise, respectively.

As Fig. 7 shows, the op-amps errors are mainly generated at the exit of the convolution module, the activation function output and the pooling output. And the impact of the SPDT noise in the pooling module also needs to be considered.

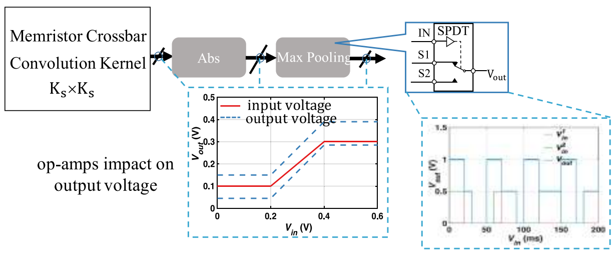

**Fig. 7.** Schematic diagram of the error location generated by op-amps and SPDTs

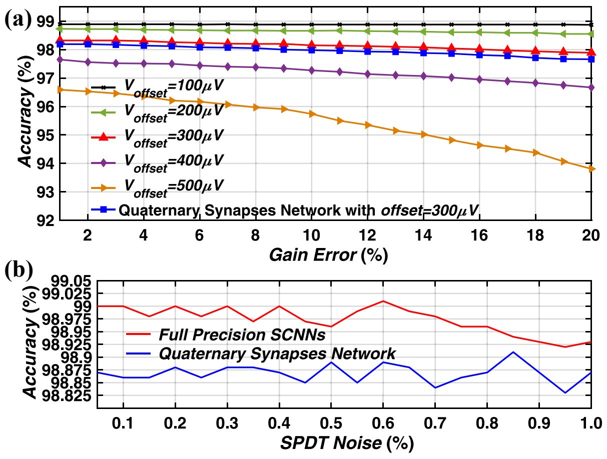

**Fig. 8.** The effect of (a) op-amps and (b) SPDTs noise on the recognition accuracy.

By combining the offset voltage and gain error of the op-amp, Fig. 8(a) demonstrates that the recognition accuracy hardly decreases when the offset voltage is lower than 400 μV. It can be seen that QSN decays by about 0.1% compared to full precision SCNNs at 300 μV op-amps offset, QSN can still achieve 97.8% recognition accuracy

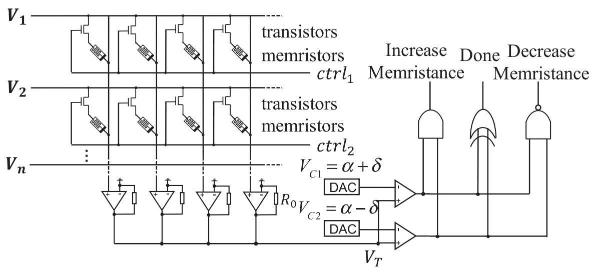

**Fig. 9.** Circuit used to program the 1T1R memristor crossbar to target memristance.

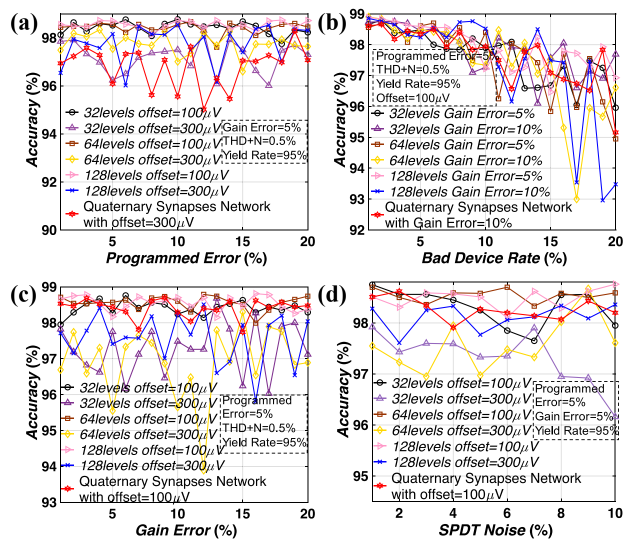

**Fig. 10.** (a) Accuracy vs. programmed error. (b) Accuracy vs. Bad Device Rate (1-yield). (c) Accuracy vs. op-amp gain error. (d) Accuracy vs. SPDT noise (THD+N).

with a gain error of 20%. Fig. 8(b) shows the impact of the SPDT noise on the recognition accuracy rate. As we can see, the SPDT noise has little effect on the recognition accuracy of the system.

Besides, the process of memristor programming should also be considered. In the simulation process, we use device resistance at different levels and randomly generate 1%–20% errors in memristor programming [31]. These errors include the error tolerance of the memristor and the errors generated by the write circuit (the write circuit is shown in Fig. 9 [32]).

Fig. 10 demonstrates the comprehensive error analysis of QSN performance. Fig. 10(a) illustrates the relationship between the recognition accuracy and programmed error. When the op-amp offset voltage is 100 µV, 95% yield and 0.5% SPDT noise, the QSN can still reach over 95% accuracy. It can be seen that the yield have the greatest impacts on the accuracy from Fig. 10(b). The QSN accuracy has a significant downward trend as the yield decreases. If the device yield is controlled above 90%, the accuracy can be kept at 97.8%. The op-amps gain error and SPDT noise have little effect on the recognition rate. The curves in Fig. 10(c) and (d) fluctuate around the mean value and there is no clear downward trend.

It can be seen that the QSN has strong robustness and is not sensitive to noise.

## 4. Conclusion

In this paper, a quaternary synapses network is proposed for low-comsumption neuromorphic memristor architecture, which maps into the only four-level memristance of memristive devices, as inspired by the recent trend of network pruning and parameter compression in the deep learning community. From the experiments above, it can be seen that the QSN provides 98.4% minimum accuracy at the interval numbers of 16, and the average accuracy can reach 98.85% which is only 0.15% decay compared with the highest accuracy. By combining the offset voltage and gain error of the op-amp, yield and programmed error of memristor crossbar array, QSN can still achieve over 95% recognition accuracy. For future work, we consider to further QSN on other neuromorphic architecture.

## Acknowledgments

This work is supported by National Natural Science Foundation of China under grant No. 61471377, No. 61604177 and No. 61704191.

## References

[1] A. Schmid: "Neuromorphic microelectronics from devices to hardware systems and applications," NOLTA **7** (2016) 468 (DOI: 10.1587/nolta.7.468).

[2] Y. Chen, *et al.*: "A novel memristor-based restricted Boltzmann machine for contrastive divergence," IEICE Electron. Express **15** (2018) 20171062 (DOI: 10.1587/elex.15.20171062).

[3] C. D. James, *et al.*: "A historical survey of algorithms and hardware architectures for neural-inspired and neuromorphic computing applications," Biologically Inspired Cognitive Architectures **19** (2017) 49 (DOI: 10.1016/j.bica.2016.11.002).

[4] Y. V. Pershin and M. Di Ventra: "Neuromorphic, digital, and quantum computation with memory circuit elements," Proc. IEEE **100** (2012) 2071 (DOI: 10.1109/JPROC.2011.2166369).

[5] Y. Taur and T. H. Ning: *Fundamentals of Modern VLSI Devices* (Cambridge University Press, Cambridge, 1998).

[6] S. Park, *et al.*: "Nanoscale RRAM-based synaptic electronics: Toward a neuromorphic computing device," Nanotechnology **24** (2013) 384009 (DOI: 10.1088/0957-4484/24/38/384009).

[7] B. C. Jang, *et al.*: "Memristive logic-in-memory integrated circuits for energy-efficient flexible electronics," Adv. Funct. Mater. **28** (2018) 1704725 (DOI: 10.1002/adfm.201704725).

[8] A. Krizhevsky, *et al.*: "Imagenet classification with deep convolutional neural networks," NIPS. Curran Associates Inc. (2012) 1097 (DOI: 10.1145/3065386).

[9] D. Dang, *et al.*: "ConvLight: A convolutional accelerator with memristor integrated photonic computing," IEEE HiPC (2017) 114 (DOI: 10.1109/HiPC.2017.00022).

[10] A. Ankit, *et al.*: "Transformer: Neural network transformation for memristive crossbar based neuromorphic system design," IEEE ICCAD (2017) 533 (DOI: 10.1109/ICCAD.2017.8203823).

[11] A. Ankit, *et al.*: "Resparc: A reconfigurable and energy-efficient architecture with memristive crossbars for deep spiking neural networks," ACM 54th Annual Design Automation Conference (2017) 27 (DOI: 10.1145/3061639.3062311).

[12] S. Sun, *et al.*: "Low-consumption neuromorphic memristor architecture based on convolutional neural networks," IEEE IJCNN (2018) 1 (DOI: 10.1109/IJCNN.2018.8489441).

[13] X. Zhu, *et al.*: "Multi-level programming of memristor in nanocrossbar," IEICE Electron. Express **10** (2013) 20130013 (DOI: 10.1587/elex.10.20130013).

[14] Z. Li, *et al.*: "Design of ternary neural network with 3-D vertical RRAM array," IEEE Trans. Electron Devices **64** (2017) 2721 (DOI: 10.1109/TED.2017.2697361).

[15] G. W. Burr, *et al.*: "Experimental demonstration and tolerancing of a large-scale neural network (165000 synapses) using phase-change memory as the synaptic weight element," IEEE Trans. Electron Devices **62** (2015) 3498 (DOI: 10.1109/TED.2015.2439635).

[16] Y. Sun, *et al.*: "A Ti/AlOx/TaOx/Pt analog synapse for memristive neural network," IEEE Electron Device Lett. **39** (2018) 1298 (DOI: 10.1109/LED.2018.2860053).

[17] H. Kim, *et al.*: "Memristor-based multilevel memory," IEEE 12th international CNNA workshop (2010) 1 (DOI: 10.1109/CNNA.2010.5430320).

[18] C. Soell, *et al.*: "Case study on memristor-based multilevel memories," Int. J. Circuit Theory Appl. **46** (2018) 99 (DOI:

10.1002/cta.2379).

[19] Y. Sun, *et al.*: "Short-term and long-term plasticity mimicked in low voltage Ag/GeSe/TiN electronic synapse," IEEE Electron Device Lett. **39** (2018) 492 (DOI: 10.1109/LED.2018.2809784).

[20] Y. L. Lecun, *et al.*: "Gradient-based learning applied to document recognition," Proc. IEEE **86** (1998) 2278 (DOI: 10.1109/5.726791).

[21] C. Szegedy, *et al.*: "Going deeper with convolutions," Proc. IEEE Conf. Computer Vision and Pattern Recognition (2015) 1 (DOI: 10.1109/CVPR.2015.7298594).

[22] K. He, *et al.*: "Deep residual learning for image recognition," Proc. IEEE Conf. Computer Vision and Pattern Recognition (2016) 770 (DOI: 10.1109/CVPR.2016.90).

[23] K. Simonyan and A. Zisserman: "Very deep convolutional networks for large-scale image recognition," arXiv:1409.1556 (2014).

[24] M. Hu, *et al.*: "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," Proc. DAC **53** (2016) (DOI: 10.1145/2897937.2898010).

[25] S. C. Wong, *et al.*: "Understanding data augmentation for classification: When to warp," arXiv:1609.08764 (2016).

[26] L. Perez and J. Wang: "The effectiveness of data augmentation in image classification using deep learning," arXiv:1712.04621 (2017).

[27] R. Takahashi, *et al.*: "Data augmentation using random image cropping and patching for deep CNNs," arXiv:1811.09030 (2018).

[28] M. Courbariaux, *et al.*: "Binaryconnect: Training deep neural networks with binary weights during propagations," Adv. Neural Inf. Process. Syst. (2015) 3123.

[29] M. Courbariaux and Y. Bengio: "BinaryNet: Training deep neural networks with weights and activations constrained to $+1$ or $-1$," arXiv:1602.02830 (2016).

[30] L. Deng, *et al.*: "Gated XNOR networks: Deep neural networks with ternary weights and activations under a unified discretization framework," Neural Netw. **100** (2017) 49 (DOI: 10.1016/j.neunet.2018.01.010).

[31] F. Alibart, *et al.*: "High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm," Nanotechnology **23** (2012) 075201 (DOI: 10.1088/0957-4484/23/7/075201).

[32] C. Yakopcic, *et al.*: "Extremely parallel memristor crossbar architecture for convolutional neural network implementation," IEEE IJCNN (2017) 1696 (DOI: 10.1109/IJCNN.2017.7966055).