Convolutional Neural Networks Based on RRAM Devices for Image Recognition and Online Learning Tasks

Zhen Dong, *Student Member, IEEE*, Zheng Zhou, Zefan Li, Chen Liu, Peng Huang, *Member, IEEE*, Lifeng Liu, Xiaoyan Liu, and Jinfeng Kang, *Senior Member, IEEE*

***Abstract*— In this paper, we devise and optimize schemes for the resistive random-access memory (RRAM)-based hardware implementation of convolutional neural networks (CNNs). The key achievements are as follows: 1) a specific circuit for CNN and corresponding operation methods is presented; 2) quantization schemes for utilizing binary RRAM or RRAM with multilevel resistances as synapses are proposed, and simulations show that our CNN system exhibits 98% accuracy on the MNIST data set using multilevel RRAM and 97% accuracy using binary RRAM as synapses; 3) the influence of factors such as the number and size of kernels, as well as the device conductance variation, on the final recognition accuracy is discussed. Three ways to reduce the hardware cost are also analyzed; and 4) we implement the online learning function using the developed CNN system with a binary spike-time-dependent plasticity protocol and achieve up to 94% accuracy for the online learning tasks.**

***Index Terms*— Convolutional neural networks (CNNs), image recognition, online learning, resistive random-access memory (RRAM), synapse.**

I. Introduction

THE use of convolutional neural networks (CNNs) has been very successful in the field of image recognition [1]–[5]. However, most software CNN algorithms are achieved by CPUs and GPUs, which are built in the traditional von Neumann architecture [6]. Deep CNNs always require a huge amount of training data (the standard training data set of ImageNet LSVRC contains 1 260 000 RGB images with size 256 × 256, the validation data set has 50 000 images, and the test data set has 100 000 images [7]), and DNN utilizes up to 10^8 weight parameters (VGG-Net utilizes 138M parameters [8]). As a result, the process of taking data from memory, sending them to CPUs and GPUs for computing, and returning the results back for storage, can become rather time-consuming and may lead to a huge hardware cost [9].

Many studies have achieved brain-inspired computing beyond the von Neumann architecture, which could realize the integration of storage and computing, and support software algorithms with high speed and low hardware cost [10]–[19]. There are generally two schemes for the hardware implementation of synapses in CNNs, which are to use CMOS circuits or novel devices like nonvolatile memory, including phase-change memory, resistive-change memory (RRAM), conductive bridge-type memory, and spin-transfer torque magnetic memory.

The RRAM-based implementation of CNNs can reduce energy consumption, improve the computational speed, and physically consider parallelism [20], [21]. However, the number of available hardware weight values depends on the available RRAM conductance states. Considering the difficulty in operating and precisely switching among a large number of conductance states, it is hard to achieve the weight precision utilized in the software algorithms (like float32 precision). Moreover, the online adjustment ΔW of synaptic weights tends to be small in the majority of algorithms, and thus is difficult to achieve in the RRAM-based implementation of CNNs due to the variation in the conductance states of RRAM devices. Though many research efforts have been made [22]–[29], a gap still exists between the software algorithm and the hardware implementation in terms of efficiently applying RRAM as synapses while maintaining recognition accuracy.

In this paper, we propose an optimized scheme for RRAM-based implementation of CNN. Sections II–IV focus on achieving an inference system. Section II introduces the CNN architecture and corresponding circuits that are used in the proposed system. The characteristics required to utilize RRAM as synapses are illustrated in Section III. In Section IV, the influences of the available hardware weight values, the number of convolutional kernels, the threshold of binary quantization scheme, and the variation of device conductance on recognition accuracy are studied in detail.

Manuscript received August 20, 2018; revised October 27, 2018; accepted November 18, 2018. Date of publication December 4, 2018; date of current version December 24, 2018. This work was supported in part by the NSFC under Grant 61334007 and Grant 61421005, in part by the National Innovation Training Program, and in part by the Beijing Municipal Science and Technology Plan Projects. The review of this paper was arranged by Editor G.-H. Koh. *(Corresponding author: Jinfeng Kang.)*

The authors are with the Institute of Microelectronics, Peking University, Beijing 100871, China (e-mail: kangjf@pku.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TED.2018.2882779

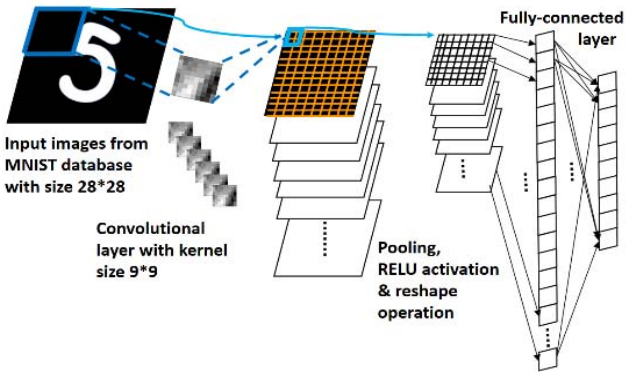


Fig. 1. Architecture of the RRAM-based CNN. The first layer is a convolutional layer, and the last layer is a fully connected layer. Between them are pooling, activation, and reshaping operations.

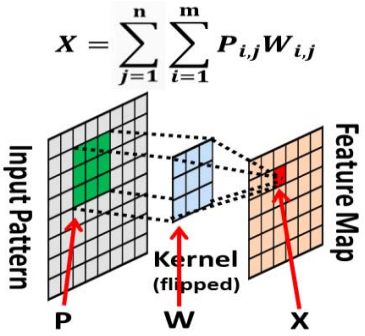


Fig. 2. Schematic of the convolution operation between an input image and a learned kernel.

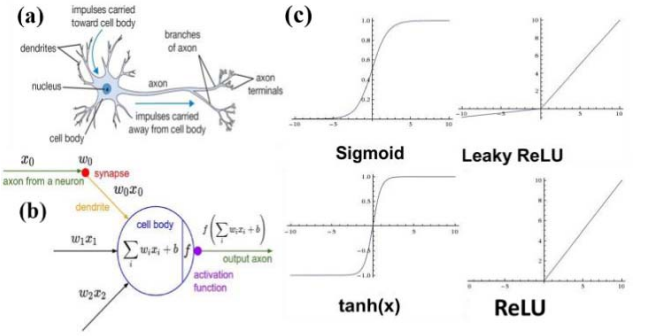


Fig. 3. (a) Schematic of a biological neuron and its components. (b) Neural model used in neural network algorithms. (c) Common types of activation functions.

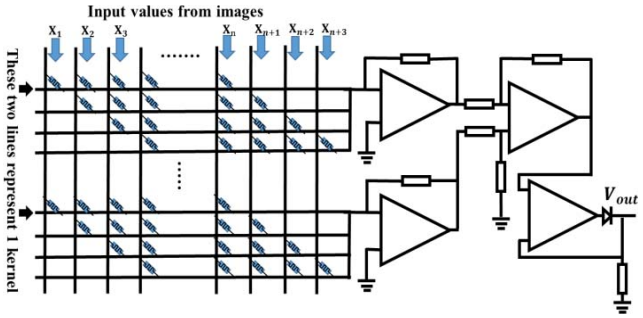


Fig. 4. Specific circuit of CNNs. Two rows of RRAM work as one kernel and eight output currents are pooled and activated together to compute one value in the output matrix, which is then conveyed to the fully connected layer.

Three ways to decrease the RRAM amount utilized in the CNN system are also discussed in Section IV. The online learning function based on the developed CNN system using a binary spike-time-dependent plasticity (STDP) protocol is achieved, and the results are presented in Section V.

II. CNN Architecture and Circuits

As shown in Fig. 1, the input images of handwritten digits from the MNIST database with size 28×28 are first applied to the convolutional layer. Then, the results are pooled and activated to generate the output features. Finally, the output features are reshaped into vectors and conveyed to a fully connected layer to make the final decision.

The definition and formula of the convolutional operation are illustrated in Fig. 2, where W is the convolutional kernel, and P is the one area in the input image that has the same size as W. X is the one pixel of the output feature map, which is equal to the sum of every product of the corresponding pixels in P and W. Therefore, one pixel in the convoluted output feature corresponds to one district in the input image, and X can become the whole feature map by moving area P.

Inspired by biological neurons, the classic neural model used in the algorithms is shown in Fig. 3, where W is the weight of the synapses, and f is the activation function. Frequently used activation functions are sigmoid, tanh, ReLU, or Leaky ReLU [30]. We utilize ReLU in this paper because it is easy for hardware implementation and is linear when the input is positive, which enables the use of RRAM conductance values that are proportional to rather than equal to software weights to represent synaptic weights in the proposed system.

Fig. 4 shows the realization of the convolutional layer with activation and pooling operations. In order to represent the negative weight values, we use the difference of currents to calculate output, namely, $I_{\text{out}} = I_a - I_b$, therefore, $W_{\text{eff}} = W_a - W_b$. ($W_{\text{eff}}$ is the effective weight.)

Traditionally, activation comes before the pooling operation. Changing the sequence can make circuit implementation easier while having little effect on the recognition accuracy. For example, according to simulations using 10 kernels with size 9×9, the recognition accuracy of the activation-first network is 98.15%. If we use the weights trained in the activation-first network while directly changing the sequence of activation and pooling, then a decent 97.98% accuracy can be achieved. Furthermore, if we train a pooling-first network from scratch, then we can obtain a 98.26% recognition accuracy, which is even higher than the original accuracy in this task.

In addition to advantages discussed earlier, utilizing ReLU activation function can also enable us to achieve a 2×2 mean pooling operation by simply combining four currents together without division circuits. As shown in Fig. 4, we can use four rows of RRAM to create four output values corresponding to four adjacent districts in the input image, namely, the blue RRAMs in the first row correspond to one selected area and

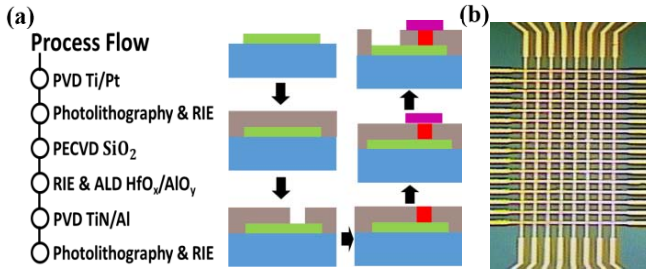


Fig. 5. (a) Process flow for the fabrication of the HfO_y-/AlO_x-based RRAM crossbar array. A schematic of each step is presented. (b) Micrograph of the fabricated 8 × 16 RRAM crossbar array.

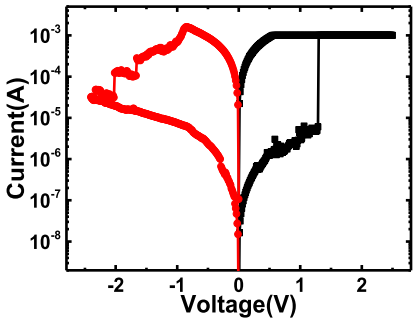


Fig. 6. Measured typical dc *I*–*V* characteristics of the RRAM device in a crossbar array.

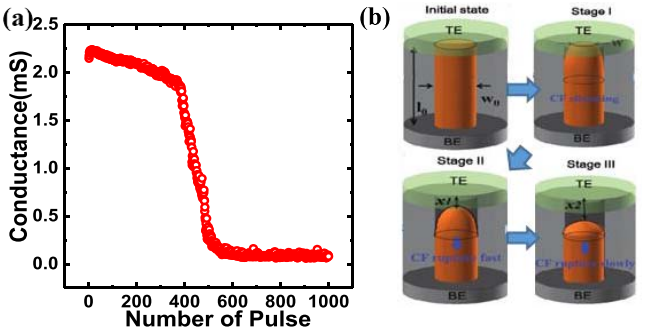


Fig. 7. (a) Conductance states obtained by adding an increasing number of small continuous ac reset pulses. (b) Schematic shape change of conductive filament corresponding to each stage in the conductance transition curve.

the blue RRAMs in the second row correspond to an adjacent area. The blank points in the RRAM crossbar array represent RRAMs that have extremely high resistance values and work as zero weights in the system.

Since infinite values cannot be realized, we actually use a clamped ReLU in Fig. 4 whose maximum value is V_{dd}. According to the data from Figs. 8 and 10, an estimation of the maximum sum of currents can be $(V_{dd} \times [0.4\ \text{mS} \times 15 + 0.6\ \text{mS} \times 15 + 1\ \text{mS} \times 5)] \times 4 \approx 0.08 \times V_{dd}$, where we assume all input signals equals V_{dd}. To avoid saturation, we can adjust pulse schemes in Fig. 8 to get multilevel states with smaller conductance, use small resistance for the first operational amplifier, or even divide input signals into groups and add the difference together to be activated. Typically, the circuit in Fig. 4 converts the pooled currents to voltages, then output 0 V or the difference of voltages if positive. The output (after a voltage follower) can serve as the input signal of the fully connected layer implemented by another RRAM crossbar array. It should be noted that lots of extra efforts are required to fulfill the design of the microarchitecture as well as auxiliary circuits which generate control signals, provide data storage, and offer I/O interfaces for our CNN systems. We believe that further optimization can be achieved by the collaboration among those related areas.

III. Device Characteristics

Fig. 5 shows the process flow for the fabrication of the proposed 8 × 16 RRAM crossbar array. First, a 20-nm Ti adhesive layer and a 100-nm Pt bottom electrode (BE) were deposited using physical vapor deposition. Then, 5 μm × 5 μm holes were formed by reactive-ion etching through the 20-nm SiO_2 isolation layer, which was deposited using plasma-enhanced chemical vapor deposition. Then, the two-layer HfO_2/Al_2O_3 structure was repeated and stacked by atomic layer deposition to form the resistive switching layer, with a total thickness of 5 nm. Finally, a 40-nm TiN and 100-nm Al top electrodes (TEs) were sputtered and patterned on the top [26].

Typical dc *I*–*V* characteristics of RRAM are tested and shown in Fig. 6, where the black curve corresponds to a set process, and the red curve corresponds to a reset process [31]. The proportion of high-resistance state (HRS) to low-resistance state (LRS), namely, the resistive window, here, is approximately 736 in Fig. 6.

In order to use a large number of available hardware weight values in the quantization scheme, we applied small reset pulses on RRAM to obtain adequate conductance states. The tested results are shown in Fig. 7(a). The schematic of the change in the conductive filament shape in the RRAM device, which is based on a physical RRAM model [32], is illustrated in Fig. 7(b). However, adding hundreds of pulses to reach wanted conductance state can be time-consuming, especially when the scale of the RRAM crossbar array is large. Thus, this operation method is used in the simulation only when there are too many conductance states needed in the quantization scheme.

Considering the efficiency of training, a feasible operation scheme among conductance states is proposed. Tested results on our RRAM array are shown in Fig. 8, where two pairs of high conductance states are measured. When adding a 1.2-V, 100-ns pulse, HCS1a will be set to a higher conductance state, namely, HCS2, which corresponds to the blue curve. HCS2 can be reset to HCS1a by applying a −1.4 V, 100-ns pulse. The operations between HCS1b and HCS3 are similar. By applying a strong reset pulse, all of the high conductance states can be reset to a low conductance of approximately 1 μS, which is small enough to be utilized as a zero weight.

Assuming that the quantization scheme uses two high-conductance states, such as HCS1a and HCS2, with the proportion of conductance equal to approximately 2.5, then, we could use weights 0, 1, and 2.5 and the difference among them (1.5) because $W_{\text{eff}} = W_a - W_b$. If the quantization

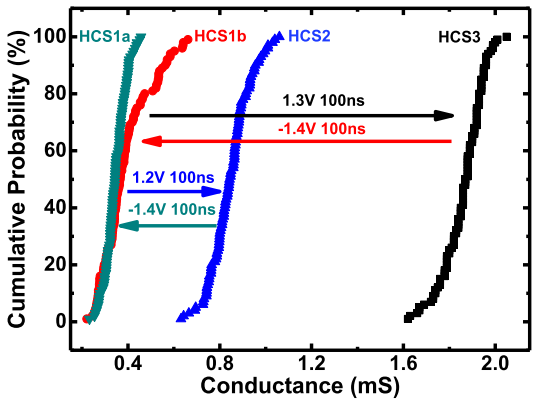


Fig. 8. Conductance states (used as quantified weights in the networks) and corresponding writing methods for RRAM devices.

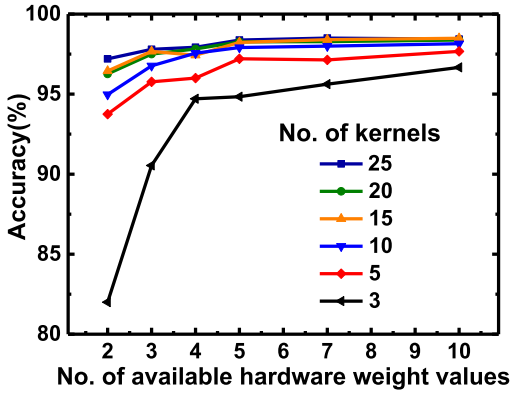


Fig. 9. Recognition accuracy as a function of the number of kernels and available hardware weight values.

scheme utilizes three high conductance states, then in addition to HCS2 and HCS3, HCS1a and HCS1b can be used as one conductance state HCS1. The proportion among those high conductance states is 1:2.3:5.2; therefore, in addition to HCS1, HCS2, and HCS3, the difference among those states, namely, 1.3, 2.9, and 4.2, respectively, can also be used by this quantization scheme.

The conductance variations of the states shown in Fig. 8 are approximately 13.4%, 28.3%, 23.5%, 10.2%, and 4.7%, corresponding to HCS1a, HCS1b, HCS1, HCS2, and HCS3, respectively. Based on our test, the variations of the lower conductance states are generally much larger than those of higher conductance states. Since the transition between HCS1b and HCS3 is greater than that of HCS1a and HCS2, the variation of HCS1b tends to be larger. However, because HCS1a and HCS1b are utilized as low weight values in the proposed CNN system, where it is actually the high weight values that play an important role, the problem of relatively large conductance variation is naturally avoided.

IV. Results and Discussion

A quantization scheme needs to map software weights to actual RRAM conductance states, which can be divided into the following two steps: 1) map software weights to available hardware weight values and 2) represent those available hardware weight values using RRAM multilevel conductance states. Since the second step has been discussed in detail in Section III, we will focus on the influence of the first step on the final performance in this section.

Fig. 9 shows the relationship between the accuracy and the number of available hardware weight values (number of $|w|$) based on RRAM with multilevel resistances. Here, the simulation results are obtained using 9×9 kernels, with mean pooling and ReLU shown in Fig. 4. If the number of available hardware weight values is larger than 4, decreasing the precision of the software weight value will not cause a considerable reduction of the recognition accuracy. Hence, the four-available-value scheme is more efficient. To obtain higher than 95% recognition accuracy, the binary RRAM scheme needs at least 10 kernels, while the four-available-value scheme needs only 5 kernels. Furthermore, according

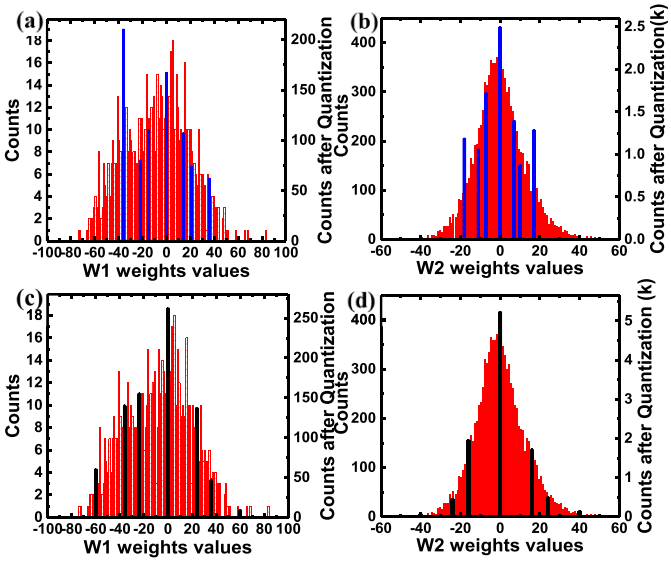


Fig. 10. Frequency histograms showing weight value distributions in (a) and (c) convolutional kernels, and (b) and (d) fully connected layer. Histograms of weight values after a small-scaling-factor quantization are shown in (a) and (b), respectively. Histograms of weights after a quantization with optimized scaling factor are presented in (c) and (d), respectively.

to Fig. 4, when the number of kernels is larger than 10, increasing the kernel number will not contribute to an increase in the recognition accuracy; therefore, using 10 kernels are considered efficient.

Fig. 10 shows the frequency histogram of software weight values in CNN with kernel size 9×9 and 10 kernels using a four-available-value scheme. When RRAMs are utilized as synapses, a quantization scheme is used to convert software values to hardware available weight values, such as 0, 1, 1.5, and 2.5 based on characteristics shown in Fig. 8. Multiplied by different scaling factors, weights of all layers can share the same quantization circuit.

Based on simulations, weights with relatively large values but small counts play an important role in the CNN system. Thus, choosing a small scaling factor and only focusing on values with large counts is inappropriate. For example, the recognition accuracy using small-scaling-factor

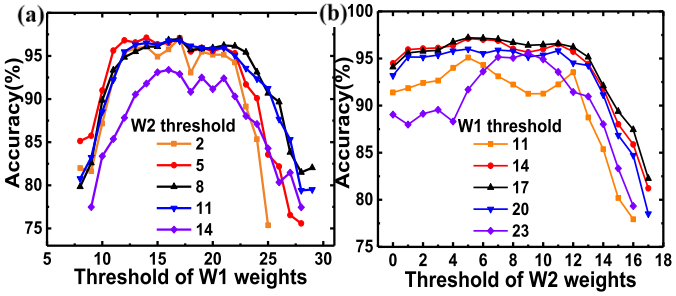


Fig. 11. (a) Influence of the W1 quantization threshold on the recognition accuracy (fixed W2 threshold). (b) Influence of the quantization threshold of W2 on the recognition accuracy (fixed W1 threshold).

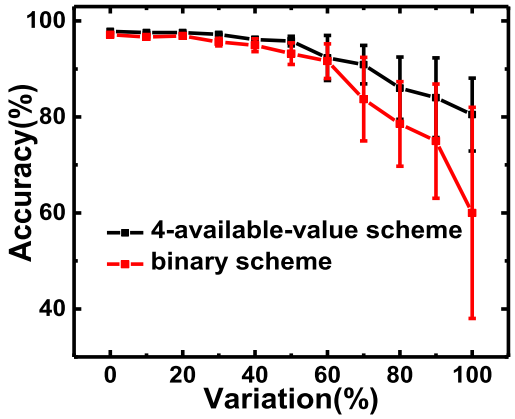


Fig. 12. Influence of the device conductance variation on the recognition accuracy.

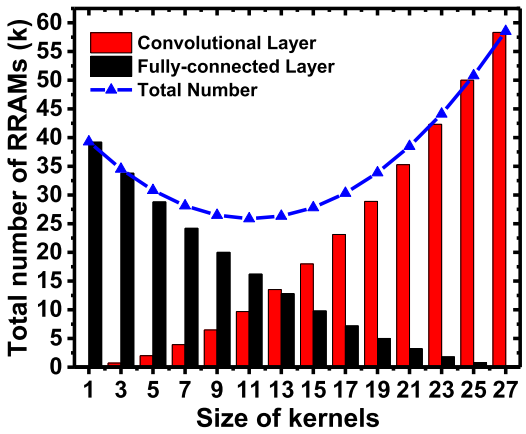


Fig. 13. Influence of the kernel size on the number of RRAMs used in the convolutional layer, the fully connected layer, as well as the whole CNN system.

quantization is 97.33%, whose results correspond to the blue lines shown in Fig. 10(a) and (b). The recognition accuracy using the quantization with optimized scaling factor is 97.92%, which corresponds to the black quantization results shown in Fig. 10(c) and (d).

Incidentally, the total number of weights in the fully connected layer (W2) is much larger than the number of weights in the convolutional layer (W1). Therefore, according to statistical laws, the distribution curve of W2 is smoother than that of W1.

The quantization threshold is crucial for the RRAM-based system using a binary scheme. The threshold determines whether a software weight value corresponds to -1, 0, or 1 and can significantly affect the recognition accuracy. Fig. 11(a) shows the influence of W1 threshold on the recognition accuracy with the threshold of W2 fixed. (W1 are weights in convolutional kernels, and W2 are weights in the fully connected layer.) In this case, 10 kernels with size 9×9 are used during simulations. Fig. 11(a) indicates that when W1 threshold is between 12 and 23, the recognition accuracy reaches a maximum of 97%. Fig. 11(b) shows the relationship between the threshold of W2 and the recognition accuracy with W1 threshold fixed, and 1–11 is the acceptable range of the W2 threshold.

Fig. 12 illustrates the system's tolerance to the variation in the device conductance. CNNs that have 10 kernels with size 9×9 are used for simulation. We use the standard deviation of measured samples as variation level to estimate the dispersion of the RRAM conductance distribution. When the variation level rises, the average accuracy begins to decrease, and the fluctuations in the recognition accuracy increase. Hence, keeping the variation, especially the variation of high conductance, lower than 50% is necessary. Fortunately, according to the measured variation of high conductance states of the HfO_y-/AlO_x-based RRAM in Fig. 8, the 50% requirement can be easily fulfilled. Furthermore, the accuracy of the binary scheme is always lower than that of the four-available-value scheme, and the fluctuation of the former tends to be larger than that of the latter in the same situation.

The above discussions mainly focus on improving the recognition performance rather than reducing the hardware cost. When the total amount of RRAM is considered a crucial criterion of the CNN system, there are generally three ways for saving RRAMs. Fig. 13 shows the relationship between the number of RRAMs and the size s, as well as the number n, of the convolutional kernels. RRAMAmount $= (s^2 \times n \times 4 + (H + 1 - s/2)^2 \times n \times 10)) \times 2$, where H is the size of input images. Note that this formula is derived using the implementation shown in Fig. 4, where four identical kernels are used together to achieve parallel convolutional computing and simplify the pooling operation. From Fig. 13 we can see that, when the kernel size is small, the cost of RRAMs in the fully connected layer is much higher than that in the convolutional layer, and the total number tends to be large. However, utilizing very large kernels could cause a huge cost of RRAMs in the convolutional layer and also reduce the recognition accuracy, which will be explained later based on Fig. 15(a). Thus, choosing a medium kernel size can make the CNN system more cost-effective.

The second way of lowering the hardware cost is to trim the original input images, which could make the output features of the convolutional layer become smaller and thus effectively reduce the number of parameters in the fully connected layer, as shown in Fig. 14. For cases using large-size kernels, since the RRAM amount used in fully connected layer is already very small, the cost reduction by trimming is not as much as in other cases.

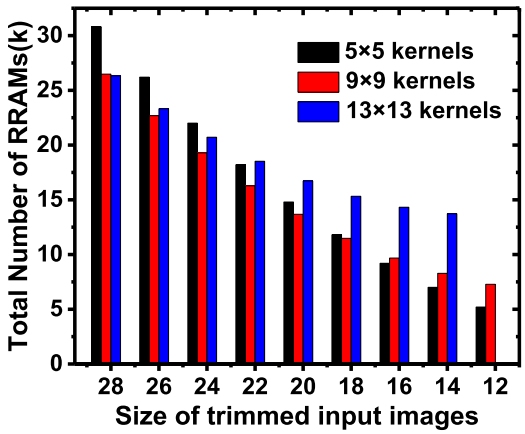


Fig. 14. Relationship between the size of trimmed input images and the total amount of RRAMs utilized in the CNN system.

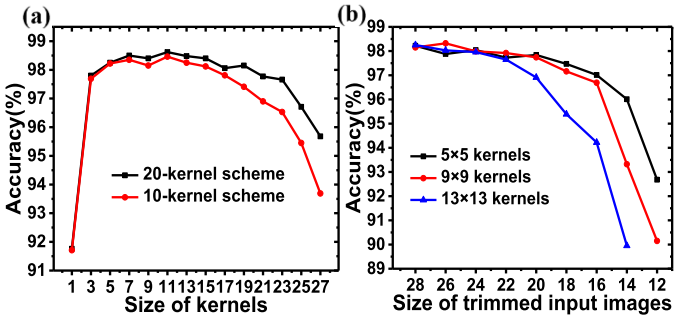


Fig. 15. (a) Influence of the kernel size and kernel number on the recognition accuracy. (b) Influence of the kernel size and the size of trimmed images on the recognition accuracy.

Fig. 15(a) and (b) presents the simulated recognition accuracies corresponding to network configurations described in Figs. 13 and 14, respectively. We can see from Fig. 15(a) that 1×1 kernels contain too few parameters for training and therefore result in low recognition accuracy. Networks with large-size kernels also cannot reach high accuracy due to the incapacity of the fully connected layer. Moreover, increasing the number of kernels can lead to more parameters for training in the fully connected layer, which partly compensates for the accuracy reduction caused by using large-size kernels. Considering both the RRAM cost and recognition performance, kernel sizes from 7×7 to 15×15 are appropriate in this specific task.

According to Fig. 15(b), the recognition accuracy will decrease quickly when the size of trimmed input images gets close to the size of convolutional kernels, due to the huge reduction of parameter counts in the fully connected layer. Moreover, the accuracy curves related to 5×5 and 9×9 kernels begin to drop before the inputs are trimmed to 16×16, namely, before the input size gets close to the kernel size. This is because the information of input images has been partly lost on account of the trimming operation.

V. Online Learning Tasks

A. Binary Operational Protocol

Hardware online learning systems based on backpropagation are difficult to achieve the following.

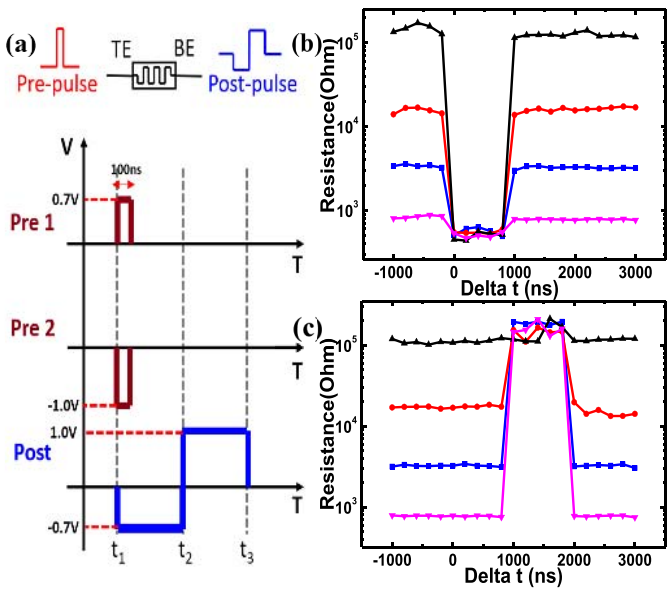


Fig. 16. (a) Waveform used in the binary STDP learning system. (b) Measured long-term potentiation characteristics of RRAM at different initial resistances. (c) Measured long-term depression characteristics of RRAM at different initial resistances. Δt is the time difference between the presignal and postsignal.

1) Complex auxiliary circuits are required to calculate results in the backpropagation.
2) Extra memory is needed to store the updated value of weights.
3) Enough conductance states are necessary for the success of weight update.
4) Multilayer backpropagation will cause more complicated calculations.

For relatively easy tasks such as the recognition of MNIST handwritten digits (compared with tasks like the ImageNet), we are able to use simpler protocols as substitutes for the backpropagation algorithm, which are easy for hardware implementation and can achieve decent performance with a much lower cost.

Fig. 16(a) shows a simple binary operational protocol, and Fig. 16(b) and (c) presents the related characteristics of the RRAM. After the forward propagation period, the fired neuron will send a feedback signal to the BE of the RRAM; then, a prepulse will be added to the TE after a time span of Δt, the length of which is determined by the input value. The input voltage larger than the threshold value will be followed by a Pre1 pulse with Δt from 0 to 1000 ns, which corresponds to a 1.4-V, 100-ns set process, while the input voltage smaller than the threshold value will be followed by a Pre2 pulse with Δt from 1000 to 2000 ns, which corresponds to a 2-V, 100-ns reset process.

Fig. 16(b) and (c) shows that, regardless of the initial RRAM resistance state, the RRAM corresponding to Pre1 would be set to an LRS of approximately 500 Ω by the superimposed signal crossing the RRAM device, and the RRAM corresponding to Pre2 would be successfully reset to an HRS of approximately 2×10^5 Ω.

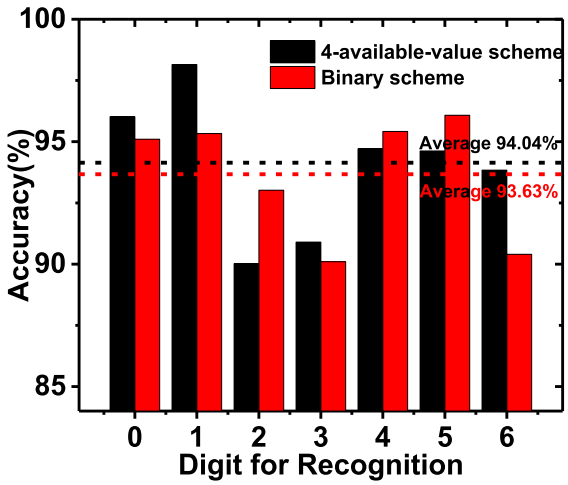


Fig. 17. Recognition results of 0–6 digits based on both the binary weight CNN and the four-available-value weight CNN. The average accuracy of each scheme is illustrated with a dotted line.

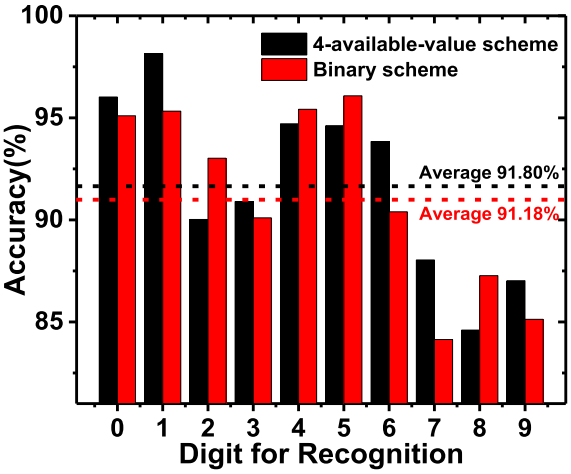


Fig. 18. Recognition results of 0–9 digits based on both the binary weight CNN and the four-available-value weight CNN. The average accuracy of each scheme is illustrated with a dotted line.

B. Online Learning Results

To solve online learning tasks, the convolutional layer copied from CNN, which uses the 10-kernel scheme to reduce the hardware cost, is fixed and used as a feature extractor. Only the fully connected layer needs to be online trained based on the average output features of the first layer and using the binary protocol. We first utilized 7000 images containing handwritten digits 0–6 selected from MNIST as the test set. The recognition accuracies of each digit based on both the binary and four-available-value CNN kernel weights are shown in Fig. 17.

The average performance using the binary convolutional weights is 93.63%, which is slightly lower than the recognition accuracy using the four-available-value scheme (94.04%).

We also present the results for all 10 000 handwritten images of the MNIST test set, as shown in Fig. 18, where 91.80% average recognition accuracy is obtained using the four-available-value scheme, and a 91.18% recognition accuracy is obtained using the binary scheme.

It should be noted that the average recognition accuracy of 10 digits is always lower than that of 7 digits, due to the difference of difficulty between those two tasks. The algorithm we proposed here is targeted to solve MNIST with the simplest operations; therefore, it may have limited expressiveness and learning capability to be used for data sets like ImageNet. In order to achieve comparable performance to backpropagation-based algorithms at the same time being applicable for hardware online learning implementation, more powerful and efficient algorithms taking full advantages of our binary protocol or other hardware-friendly learning protocols (which may need multilevel or analog RRAMs) need to be explored and devised, which is the emphasis of our work in the future.

VI. Conclusion

The CNN architecture and related circuits are optimized in this paper. The multilevel resistance states of RRAM are measured and utilized in an RRAM-based implementation of CNN. Different factors that affect recognition accuracy are analyzed in detail. Three ways to reduce the hardware cost are proposed. Finally, online learning tasks are accomplished based on the developed CNN system.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[2] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May/Jun. 2010, pp. 253–256.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] D. Silver, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[6] F. Akopyan *et al.*, "TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 34, no. 10, pp. 1537–1557, Oct. 2015.

[7] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2014.

[8] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[9] J. von Neumann, *The Computer and the Brain*. New Haven, CT, USA: Yale Univ. Press, 2012.

[10] P. Kinget and M. S. J. Steyaert, "A programmable analog cellular neural network CMOS chip for high speed image processing," *IEEE J. Solid-State Circuits*, vol. 30, no. 3, pp. 235–243, Mar. 1995.

[11] G. Indiveri, E. Chicca, and R. Douglas, "A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity," *IEEE Trans. Neural Netw.*, vol. 17, no. 1, pp. 211–221, Jan. 2006.

[12] V. Milo *et al.*, "Demonstration of hybrid CMOS/RRAM neural networks with spike time/rate-dependent plasticity," in *IEDM Tech. Dig.*, Dec. 2016, pp. 16.8.1–16.8.4.

[13] K.-H. Kim *et al.*, "A functional hybrid memristor crossbar-array/CMOS system for data storage and neuromorphic applications," *Nano Lett.*, vol. 12, no. 1, pp. 389–395, 2011.

[14] R. Z. Han *et al.*, "A novel convolution computing paradigm based on NOR flash array with high computing speed and energy efficient," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–4.

[15] S. B. Eryilmaz, D. Kuzum, S. Yu, and H.-S. P. Wong, "Device and system level design considerations for analog-non-volatile-memory based neuromorphic architectures," in *IEDM Tech. Dig.*, Dec. 2015, pp. 4.1.1–4.1.4.
[16] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, and H.-S. P. Wong, "A low energy oxide-based electronic synaptic device for neuromorphic visual systems with tolerance to device variation," *Adv. Mater.*, vol. 25, no. 12, pp. 1774–1779, 2013.
[17] G. W. Burr, "Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element," *IEEE Trans. Electron Devices*, vol. 62, no. 11, pp. 3498–3507, Nov. 2015.
[18] M. Suri *et al.*, "Bio-inspired stochastic computing using binary CBRAM synapses," *IEEE Trans. Electron Devices*, vol. 60, no. 7, pp. 2402–2409, Jul. 2013.
[19] B. Gao, H. Wu, J. Kang, H. Yu, and H. Qian, "Oxide-based analog synapse: Physical modeling, experimental characterization, and optimization," in *IEDM Tech. Dig.*, Dec. 2016, pp. 7.3.1–7.3.4.
[20] T. Gokmen, M. Onen, and W. Haensch, "Training deep convolutional neural networks with resistive cross-point devices," *Frontiers Neurosci.*, vol. 11, p. 538, Oct. 2017.
[21] I.-T. Wang, Y.-C. Lin, Y.-F. Wang, C.-W. Hsu, and T.-H. Hou, "3D synaptic architecture with ultralow sub-10 fJ energy per spike for neuromorphic computation," in *IEDM Tech. Dig.*, Dec. 2014, pp. 28.5.1–28.5.4.
[22] F. Alibart, E. Zamanidoost, and D. B. Strukov, "Pattern classification by memristive crossbar circuits using *ex situ* and *in situ* training," *Nature Commun.*, vol. 4, p. 2072, Jun. 2013.
[23] S. Park *et al.*, "RRAM-based synapse for neuromorphic system with pattern recognition function," in *IEDM Tech. Dig.*, Dec. 2012, pp. 10.2.1–10.2.4.
[24] D. Garbin *et al.*, "HfO_2-based OxRAM devices as synapses for convolutional neural networks," *IEEE Trans. Electron Devices*, vol. 62, no. 8, pp. 2494–2501, Aug. 2015.
[25] J. Woo, K. Moon, J. Song, M. Kwak, and J. Park, "Optimized programming scheme enabling linear potentiation in filamentary HfO_2 RRAM synapse for neuromorphic systems," *IEEE Trans. Electron Devices*, vol. 63, no. 12, pp. 5064–5067, Dec. 2016.
[26] Z. Zhou *et al.*, "The characteristics of binary spike-time-dependent plasticity in HfO_2-based RRAM and applications for pattern recognition," *Nanosc. Res. Lett.*, vol. 12, no. 1, p. 244, 2017.
[27] S. Yu *et al.*, "Binary neural network with 16 Mb RRAM macro chip for classification and online training," in *IEDM Tech. Dig.*, Dec. 2016, pp. 16.2.1–16.2.4.
[28] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, pp. 61–64, 2015.
[29] S. G. Hu *et al.*, "Associative memory realized by a reconfigurable memristive Hopfield neural network," *Nature Commun.*, vol. 6, p. 7522, Jun. 2015.
[30] F.-F. Li, A. Karpathy, and J. Johnson, "CS231n: Convolutional neural networks for visual recognition," Stanford Univ., Stanford, CA, USA, Tech. Rep., Jan. 2016.
[31] Z. Dong *et al.*, "RRAM based convolutional neural networks for high accuracy pattern recognition and Online learning tasks," in *Proc. Silicon Nanoelectron. Workshop (SNW)*, Jun. 2017, pp. 145–146.
[32] P. Huang *et al.*, "Compact model of HfO_x-based electronic synaptic devices for neuromorphic computing," *IEEE Trans. Electron Devices*, vol. 64, no. 2, pp. 614–621, Feb. 2017.

Zhen Dong (S'16) is currently pursuing the B.S. degree with the Institute of Microelectronics, Peking University, Beijing, China.

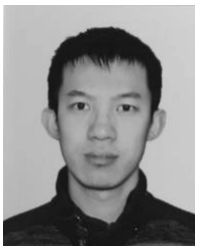
Since 2016, he has been a Research Assistant with Stanford University, Stanford, CA, USA, where he is currently an Undergraduate Visiting Researcher. His current research interests include hardware neuromorphic systems, deep learning accelerators, and nonvolatile memories.

Zheng Zhou received the B.E. degree from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2013. He is currently pursuing the Ph.D. degree with the Institute of Microelectronics, Peking University, Beijing, China.

His current research interests include resistive random-access memory and neuromorphic computing.

Zefan Li is currently pursuing the B.S. degree with the Institute of Electronics, Peking University, Beijing, China.

His current research interests include non-von Neumann computing based on resistance random-access memory.



Chen Liu received the B.S. degree from Peking University, Beijing, China, in 2014, where he is currently pursuing the Ph.D. degree with the Institute of Microelectronics.

Peng Huang (M'16) received the B.S. degree from Xidian University, Xi'an, China, in 2010 and the Ph.D. degree in microelectronics from Peking University, Beijing, China, in 2015.

He is currently a Post-Doctoral Associate with the Institute of Microelectronics, Peking University. His current research interests include resistive random-access memory and its application in computing.

Lifeng Liu received the Ph.D. degree in materials physics and chemistry from the Institute of Semiconductors, Chinese Academy of Sciences, Beijing, China, in 2005.

He is currently an Associate Professor with the Institute of Microelectronics, Peking University, Beijing.

Xiaoyan Liu received the B.S., M.S., and Ph.D. degrees in microelectronics from Peking University, Beijing, China, in 1988, 1991, and 2001, respectively.

She is currently a Professor with the Institute of Microelectronics, Peking University.

Jinfeng Kang (SM'16) received the Ph.D. degree in solid-state electronics from Peking University, Beijing, China, in 1995.

He is currently a Professor with the Institute of Microelectronics, Peking University.