SCIENCE CHINA
Information Sciences

June 2021, Vol. 64 160406:1–160406:9
https://doi.org/10.1007/s11432-020-3198-9

• RESEARCH PAPER •
Special Focus on Near-memory and In-memory Computing

Array-level boosting method with spatial extended allocation to improve the accuracy of memristor based computing-in-memory chips

Wenqiang ZHANG, Bin GAO*, Peng YAO, Jianshi TANG, He QIAN & Huaqiang WU

Institute of Microelectronics, Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100084*, China.*

Received 31 December 2020/Accepted 25 February 2021/Published online 21 April 2021

Abstract Memristor based computing-in-memory chips have shown the potentials to accelerate deep neural networks with high energy efficiency. Due to the inherent filament-based conductive mechanism of the memristor, the reading and writing noises are hard to eliminate. Besides, the precision of the large-scale memristor array is still limited. However, when the noise of the memristor is large, the existing training methods to reduce the accuracy loss of memristor based computing-in-memory chips will face challenges. Hence, we proposed the array-level boosting method with spatial extended allocation to reduce the accuracy loss induced by the limited precision and large noises. To optimize the spatial allocation number of each layer in the neural network, the greedy spatial extended allocation algorithm is also proposed. The image processing and classification tasks are demonstrated based on fabricated 32×128 memristor arrays to valid the performance of the proposed method. The chip-in-loop results show that the recovered accuracy of ResNet-34 on CIFAR-10 with array-level boosting method is 92.3%, which is closed to software-based accuracy of 93.2%.

Keywords memristor, computing-in-memory, array-level boosting, neuromorphic computing, RRAM

Citation Zhang W Q, Gao B, Yao P, et al. Array-level boosting method with spatial extended allocation to improve the accuracy of memristor based computing-in-memory chips. Sci China Inf Sci, 2021, 64(6): 160406, https://doi.org/10.1007/s11432-020-3198-9

1 Introduction

Over the past several years, deep neural networks (DNNs) based artificial intelligence (AI) has extensive promising transformative applications that exceeded human-level capabilities in many areas, such as image detection and recognition [1, 2], and machine translation [3]. Substantial complementary metal-oxide-semiconductor transistor (CMOS) based edge DNN accelerators [4] have emerged to promote these applications to the power-hungry internet-of-things hardware. These chips are flexible that can support versatile neural network models. Whereas, when improving the energy efficiency of these chips, challenges come out that the scaling down of device technology is gradually slow [5] and the massive data need to be exchanged between computing and memory units [6].

To deal with these fundamental limitations of existing DNN accelerators, the emerged memory-based computing-in-memory chip for DNNs has been investigated as a promising candidate [7]. Computing-in-memory chips improve the energy efficiency by executing computing operations locally within memory units based on the physical laws and reducing data transmission to a minimum between computing and memory units [8]. Both charge-based memory such as static random-access memory (SRAM) [9, 10] and floating gate field effect transistor (FGFET) [11], and two/three terminal resistance-based memory such as resistive random-access memory (RRAM) [12, 13], phase-change memory (PCM) [14, 15] and ferroelectric field effect transistor (FeFET) [16] have been introduced into computing-in-memory chips as basic computing units. For the resistance-based memory in the computing-in-memory application, the

* Corresponding author (email: gaob1@tsinghua.edu.cn)

devices are arranged as crossbar arrays. When the conductance matrix is mapped to the array and the voltage vector is parallelly applied to the row terminals, the accumulated output current vector can be generated based on the Ohm's Law and Kirchhoff's Law, which is the computing result of vector-matrix multiplication [8]. Array-level and macro-level analog RRAM (also called memristor) based chips have demonstrated many applications such as letter/image classification [17,18], face recognition [19] and time series prediction [20], which have shown the potentials in large-scale integration and practical deployment.

However, due to the nonidealities of memristor devices, especially reading and writing noises, the system accuracy of large-scale memristor based computing-in-memory chips will drop. Many systematic methods have been proposed to reduce the effects of noises based on simulations and experiments. There are mainly three types of methods to achieve the noise-tolerate neural network: off-chip noise aware training, on-chip training, and local multiple-device representation. The noise aware training leverages the noise injection to weights in the forward phase and uses a straight-through estimation method to calculate the residual and gradient of weights in the backward phase [21, 22]. Whereas, the constraint of this method is that the on-chip accuracy will drop a lot when the variation of memristor devices is high. The on-chip training utilizes the delicately designed backpropagation circuits to perform the on-chip calculation of gradient and local update of the memristor array [23, 24]. This method requires extra backpropagation circuits, resulting in the heavy overhead of area and energy. Besides, the nonideal fluctuations of memristor devices in programming operations will make the updated gradients deviate from the proper values. The neural network with multiple memristor devices as one weight [25] can reduce the variance of weights by compensating intra-device noise, and the committee machine with multiple neural networks employed ensemble averaging to improve the accuracy [26]. However, the accuracy drop effects of different layers in the neural network by nonidealities are varied, and the universal common multiple and averaging method from pre-layers to post-layers results in the overhead of memristor usage. Besides, the local integration of multiple memristor devices as one weight within the same array lacks flexibility to utilize the various accuracy importance of different layers.

In this work, we proposed an array-level boosting method by integrating the spatial extended allocation for memristor based computing-in-memory chip to compensate quantized and intrinsic noises of memristors. The array-level boosting method leverage the flexible combination of multiple arrays to deal with different accuracy importance of different layers. Here, we firstly present the fabricated 32×128 memristor array and show the measured multi-level characteristics. Then, we show the boosting results of the image processing application. Further, we show the performance of the array-level boosting method on ResNet-34 with CIFAR-10 dataset and discuss the results.

2 Method

The DNN basically consists of convolutional layers, fully-connected layers, pooling layers and batch normalization layers. The ResNet family has shown excellent performance on many computing vision tasks [2], which is a widely used neural network architecture. Thus, we employed a middle-size neural network in ResNet family — ResNet-34 as the example in this work. ResNet-34 is a cascade of a single convolutional layer, three residual blocks, a global average pooling layer and a fully-connected layer (Figure 1(a)). Three residual blocks have ten, twelve and ten convolutional layers, respectively. The output of each two layers in residual blocks is connected with the input with an element-wise add operation. The most computing-intensive layers of DNNs are convolutional layers and fully-connected layers, which are suitable for memristor based computing. The pooling layer can be calculated with digital computing units, while the batch normalization layer can be fused into the convolutional layer by transforming the weights of the convolutional layer. Then, we mainly consider the convolutional layer and fully-connected layer. The convolutional layer can be formulated as

$$\mathrm{FM}_{\mathrm{out}}(x,y,z)=\sum_{i=0}^{K-1}\sum_{j=0}^{K-1}\sum_{k=0}^{C_{\mathrm{in}}-1}\mathrm{FM}_{\mathrm{in}}(x+i,y+i,k)\mathrm{Kernal}_z(i,j,k), \tag{1}$$

where i, j, and k are the spatial coordinate of the convolutional kernel, $\mathrm{FM}_{\mathrm{in}}$ $(C_{\mathrm{in}},H_{\mathrm{in}},W_{\mathrm{in}})$ and $\mathrm{FM}_{\mathrm{out}}$ $(C_{\mathrm{out}},H_{\mathrm{out}},W_{\mathrm{out}})$ are input and output feature maps, respectively. C_{in}, C_{out}, and K are input channel, output channel, and kernel size, respectively. To use the memristor array to perform the calculation of convolutional layer, the four-dimension convolutional kernel $(C_{\mathrm{out}},C_{\mathrm{in}},K,K)$ is unrolled to two-dimension

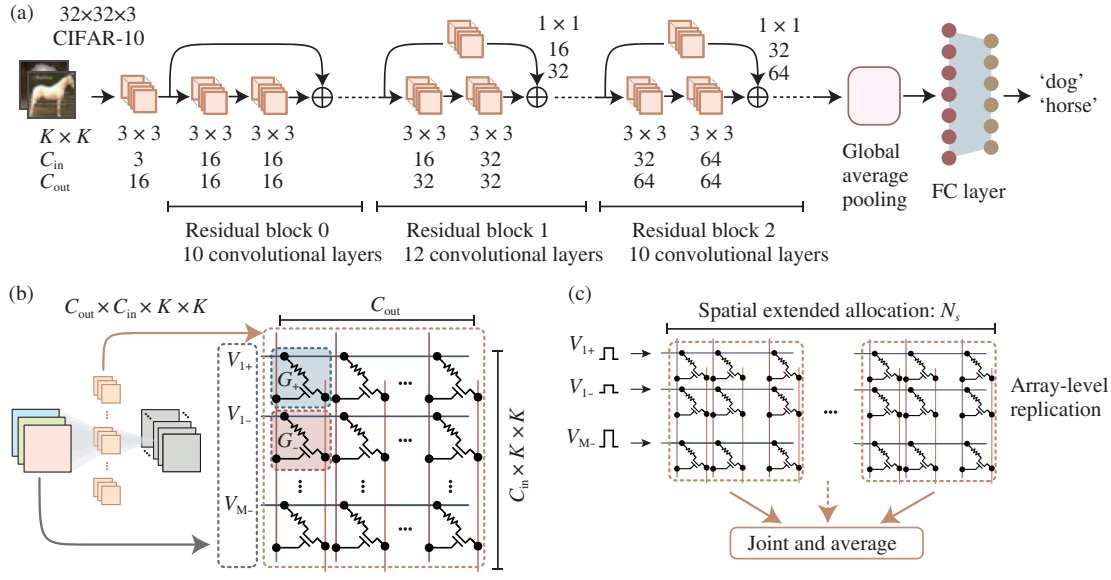

Figure 1 (Color online) Array-level boosting method. (a) The architecture of ResNet-34. (b) The principles of convolutional operation with the memristor array. The convolutional kernel is flattened from $(C_{\text{out}}, C_{\text{in}}, K, K)$ to $(C_{\text{out}}, C_{\text{in}} \times K \times K)$ and mapped to the memristor array with differential rows. (c) The schematic of the array-level boosting method with spatial extended allocation.

matrix $(C_{\text{out}}, C_{\text{in}} \times K \times K)$ (Figure 1(b)). After the input feature map is sequentially transformed to voltage vector and applied to the input memristor array, the output feature map can be generated by reconstruction from the output current vector. Here, we use a pair of two 1-transistor-1-memristor (1T1R) cells (g_+, g_-) within the conductance window $(g_{\min}, g_{\max})$ to represent one synaptic weight (w). The synaptic weight is linearly mapped to the differential memristors $(w \to g = g_+ - g_-)$ where

$$g_+ = \begin{cases} \dfrac{w}{w_{\max}} \times (g_{\max} - g_{\min}) + g_{\min}, & w > 0, \\ g_{\min}, & w \leqslant 0, \end{cases} \tag{2}$$

$$g_- = \begin{cases} g_{\min}, & w \geqslant 0, \\ \dfrac{-w}{w_{\max}} \times (g_{\max} - g_{\min}) + g_{\min}, & w < 0, \end{cases} \tag{3}$$

and $w_{\max}$ is the maximum of the weights in this convolutional layer. The fully-connected layer is similar to convolutional layer except that the weight does not need to be unrolled and can be directly mapped to the memristor array.

As shown in Figure 1(c), the array-level boosting method introduces the spatial extended allocation of arrays to improve the precision of memristor based computing-in-memory chips. When considering the quantization error, reading and writing noises of multi-level memristor, the output current of memristor array can be expressed as

$$I_{\text{out}} = (G + q_g + n_{\text{rw}})V_{\text{in}}, \tag{4}$$

where q_g is the quantization error of memristor array, n_{rw} is the reading and writing noise of memristor array. When the memristor is programmed to a specify conductance level g, this conductance level can be represented as Gaussian distribution with conductance-related standard deviation, which can be formulated as $\sigma(g) = f(g)$, where f usually is a polynomial function [21]. The standard deviation of output noise due to the write and read noise of memristor is $\sigma_{\text{out}}^w = \sqrt{\Sigma_0^{N-1} f^2(g_i)}$. Thus, to reduce the standard deviation of output noise, the memristor array can be replicated by N_s times and calculated the average of the output. The standard deviation of replication and average output is $\sigma_{\text{out}}^w/\sqrt{N_s}$. Besides, when the conductance window of the memristor is $(g_{\min}, g_{\max})$ and the memristor is N_g bits, the q_g can be represented with a uniform distribution as $U(-\frac{g_{\max}-g_{\min}}{2^{N_g}-1}, \frac{g_{\max}-g_{\min}}{2^{N_g}-1})$ and the standard deviation of quantization error of a computing cell is $\sigma^q = (\frac{g_{\max}-g_{\min}}{2^{N_g+\log_2 N_s}-1})^2/12$ with add operation. Unlike that the

existing bit splicing method will amplify the noise of most significant bits (MSBs), the array-level spatial extended allocation method can reduce the effects of noise with add and average. Then, with the spatial extended allocation that the memristor array can be replicated by N_s times, both of the quantization error and write noise can be reduced.

To determine the N_s of each layer (N_s^l) in the neural network, we need to solve the problem: $\text{argmax}_{N_s^l, l\in(1,L)} |\text{Acc}_b - \text{Acc}|$. Besides, the extended allocation of arrays should as small as possible. However, the search space of N_s is $\Pi_{l=1}^{L} N_s^l$, which is too large to get the optimized results by linearly searching when the neural network is deep. Thus, we proposed the greedy spatial extended allocation (GSEA) algorithm to find the appropriate N_s of each layer, as shown in Algorithm 1. In GSEA, we first heuristically select the accuracy threshold A_{th} and the maximum spatial extended allocation N_s^m. A_{th} is used to determinate the convergence of the GSEA algorithm. N_s^m is the maximum of array-level spatial extended allocation to limit the overhead of GSEA algorithm. The ideal accuracy (Acc) is calculated and the N_s^l of each layer is initialized to zero. Then, GSEA algorithm will optimize the N_s^l from the first layer to the last layer. For each layer, the accuracy difference $|\text{Acc}_b - \text{Acc}|$ is calculated to compare with A_{th}. When the accuracy difference is larger than A_{th}, the N_s^l will be updated. Note that when optimizing the N_s of the l^{th} layer, the N_s of other layers are set to one.

Algorithm 1 Greedy spatial extended allocation method

Require: Neural network, dataset, accuracy threshold A_{th}, maximum spatial extended allocation N_s^m.
Ensure: Spatial extended allocation of each layer: N_s^l.

```
Calculate the accuracy of neural work without noise: Acc;
Initialize N_s^l, l ∈ (1, L);
for l = 1 to L do
    while |Acc_b − Acc| > A_th do
        N_s^l = N_s^l + 1;
        if N_s^l > N_s^m then
            N_s^l = N_s^m;
        end if
        Calculate the accuracy with N_s: Acc_b;
    end while
end for
return N_s^l, l ∈ (1, L).
```

3 Results and discussion

3.1 Device and array characteristics

In order to validate the performance of array-level boosting with GSEA algorithm, we develop an experimental verification system with fabricated memristor arrays (Figure 2(a)). The array has 4096 1T1R cells which are arranged as 32×128. To utilize the parallelism of the array, the 32 source lines (SLs) is perpendicular to the 128 bit lines (BLs) and parallel to 32 word lines (WLs), which is suitable for the neural network application that the dimension of input is usually larger than that of output. To measure the 32×128 memristor array, we develop a customized test system that can apply the voltages and sense the currents to the arbitrary terminals parallelly. In Figure 2(a), the 1T1R array employed backend of line (BEOL) fabrication technology that the memristor is fabricated after the transistor is fabricated from the foundry. To improve the multi-level characteristics of memristor, we adopted a thermal enhanced layer (TEL) in the TiN/TEL/HfO_x/TiN material stack [27].

The conductance window of the memristor is linearly divided into eight level range from 2 to 20 μS. In Figure 2(b), the memristor device in 32×128 memristor array is programmed with the close-loop mapping strategy to eight levels, and we sweep the read voltage from 0 to 0.2 V with 5 V as the gate access voltage. As shown in Figure 2(b), the measured current of the memristor is approximately in direct proportion to the read voltage, thus, the input data in the following neural network application is encoded with the varied amplitude of voltage from 0 to 0.2 V. In the following measurements of device characteristics, the read voltage is set to 0.2 V. Due to the inherent filament based conductive mechanism of memristor, the measured conductance at each level will have noise and fluctuation (Figure 2(c)). For the write noise, 512 memristors in the first 16 columns of 32×128 memristor array are sequentially programmed into eight levels with 0.37 μA as the relative programming error boundary. The cumulative current distribution of each level is presented in the upside inset of Figure 2(d). To quantify programming error of each level,

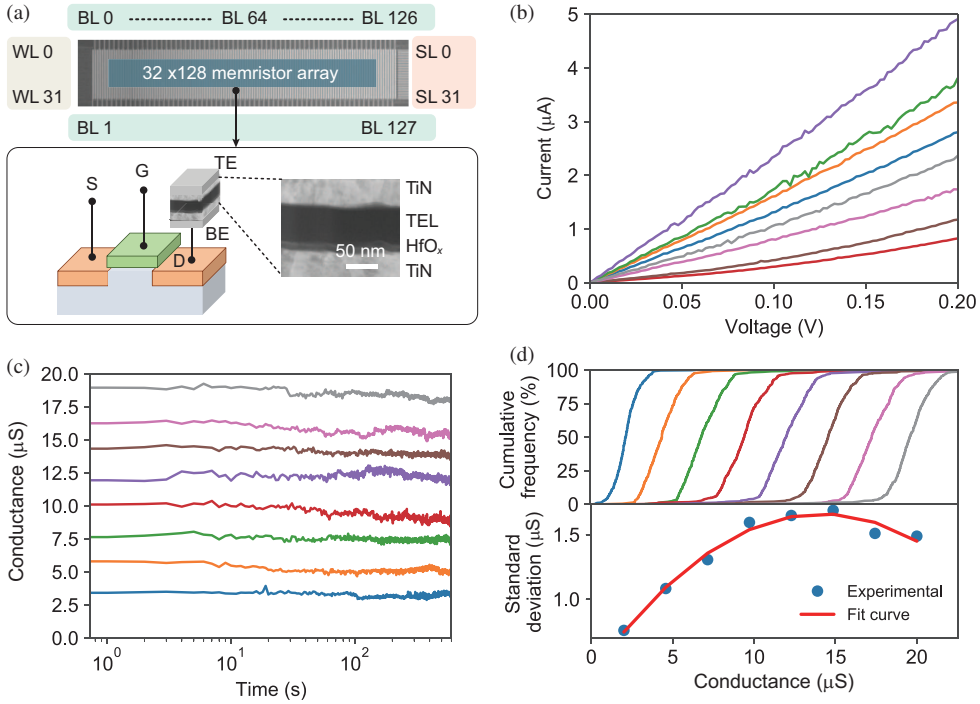

Figure 2 (Color online) Experimental characteristics of fabricated memristor array. (a) Structure of device and 32×128 array; (b) and (c) current voltage relation and read noise of conductance when the device is programmed to eight different conductance levels; (d) the cumulative distribution and corresponding standard deviation of eight different conductance levels.

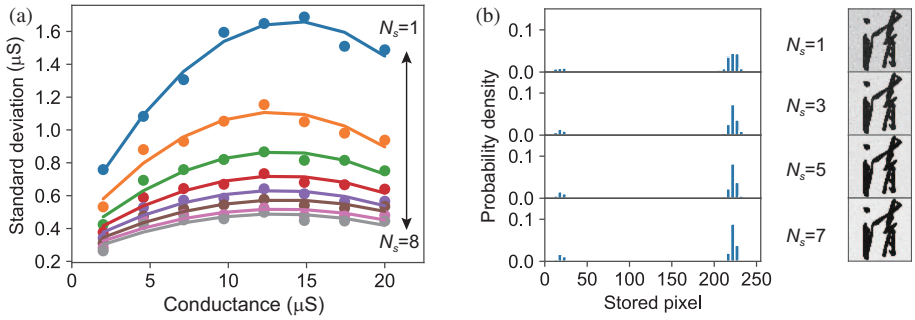

Figure 3 (Color online) Array-level boosting of data representation. (a) The standard deviation of eight conductance levels under the different N_s; (b) the distribution of programmed images with different N_s.

the standard deviation of each level is calculated (bottom inset of Figure 2(d)), which can be fitted with a quadratic function. The average standard deviation is about 8% of the current window. In the following part, the array-level boosting method will be deployed to the image processing and classification tasks and performance of it will be evaluated.

3.2 Array-level boosting for image processing

In an image processing application, the transformation matrix is stored in the memristor array and the performance of image processing is sensitive to the variation. Thus, we use the image processing application to show that the array-level boosting method can decrease the variation of programmed multi-level data. We first present the transition of different level distributions from low conductance to high conductance. The 32×128 memristor array is logically divided into eight subarrays and each subarray has 16 columns as well as 512 memristor devices. Then each subarray is temporally programmed to eight different levels and calculates the standard deviation of each subarray and each level. As shown in Figure 3(a), when the N_s is changed from 1 to 8, the largest standard deviation among eight levels decreases from about 0.3 μA to about 0.1 μA. Besides, the trend of standard deviation declining in the transition of N_s from 1 to 4 is faster than that in transition of N_s from 4 to 8 (0.2 μA vs. 0.05 μA), which is not

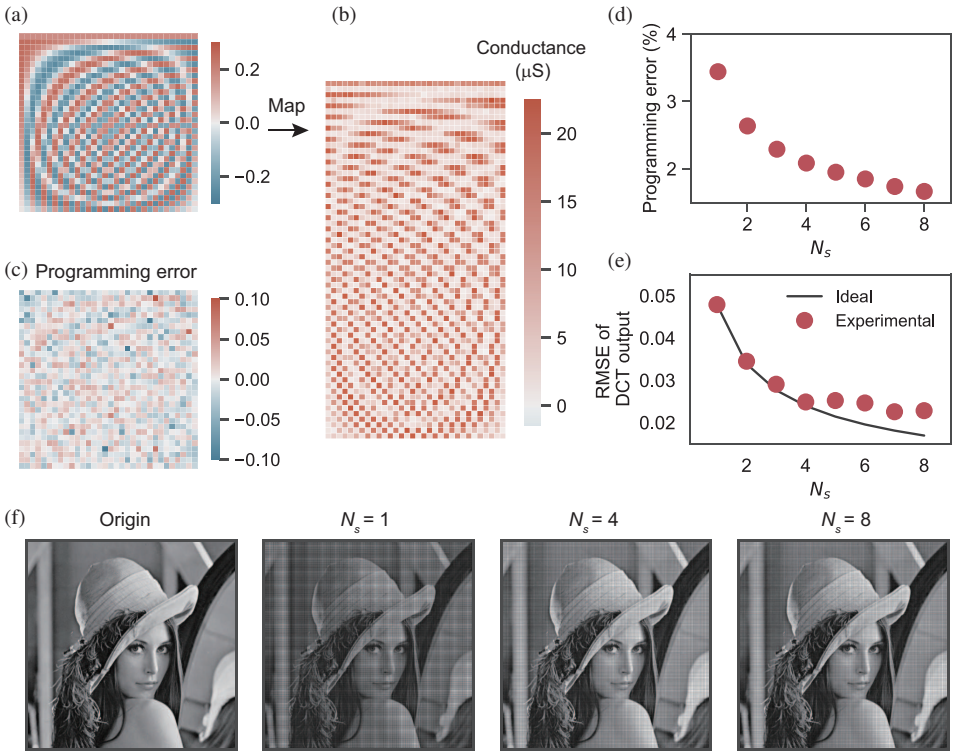

Figure 4 (Color online) Array-level boosting for the image processing application. (a) The origin discrete cosine transformation matrix; (b) the read currents of 32×128 array after mapping and programming of origin discrete cosine transformation matrix; (c) the programming error matrix of (b); (d) and (e) the trajectories of related average programming error and discrete cosine transformation output root-mean-square error when the N_s is changed; (f) the transition of images after discrete cosine transformation and reverse discrete cosine transformation when using different N_s.

efficient to replicate the array with too many times. A similar but more visualized example has been presented in Figure 3(b). When the word image is programmed into the array, the near white part of the gray image is filled with visible noise. In Figure 3(b), as the N_s is changed from 1 to 7, the near white noise is reduced.

Then, we show that the array-level boosting method can reduce the variation of output in discrete cosine transformation (DCT). The DCT calculation can be expressed as

$$F = AfA^{\mathrm{T}}, \tag{5}$$

where A is discrete cosine transformation matrix which can be calculated by $c(i)\cos(\frac{(2j+1)\pi}{2N}i)$. The $c(i)$ is $\sqrt{1/i}$ when $i = 0$ and $\sqrt{2/i}$ when $i \neq 0$. f and F are the input image and the frequency spectrum, respectively. Here, to fit the A with the size of the fabricated memristor array, the dimension of A is set to 32 (Figure 4(a)). Similar to the differential representation of weight for neural networks as elaborated in Section 2, each element in DCT matrix is mapped into two memristor devices in adjacent rows (Figure 4(b)). In the physical implementation, the voltage inputs are parallelly applied to the row terminals of the crossbar and the output results are generated by accumulating the current of each row. Compared to the origin A with the reconstructed matrix from programmed currents on the memristor array, the programming error can be generated in Figure 4(c). As shown in Figures 4(d) and (e), when the N_s is linearly increased, the average programming error of each cell and the root-mean-square error (RMSE) of DCT output will decrease super-linearly. In the ideal situation, the output RMSE should decrease by $1/\sqrt{N_s}$ when the N_s increased. For the trajectory of output RMSE, the decreasing trend fits well with the ideal decreasing trend when N_s is less than 4. When N_s is larger than 4, the output RMSE is saturated to a certain value. This can owe to the fact that the distribution of values in the input image and differential programmed current matrix of A are not the ideal Gaussian distribution, and thus the accumulation of programming and computing error cannot be counteracted. The origin input image and the recovered image based on DCT and inverse DCT are shown in Figure 4(f). When the N_s is 1, the

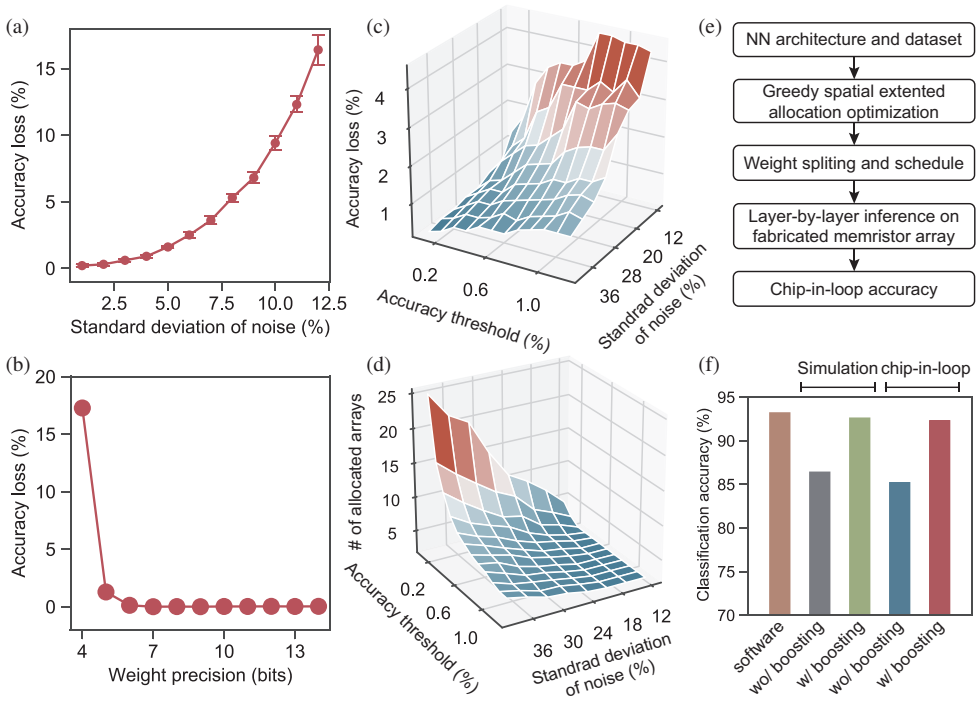

Figure 5 (Color online) The results of the array-level boosting method with greedy spatial extended allocation on ResNet-34. (a) and (b) The simulated accuracy loss under different standard deviations of reading and writing noise, and weight precision; (c) the optimized accuracy and (d) allocated arrays when the accuracy threshold and standard deviation of noise are different; (e) the diagram of chip-in-loop emulation; (f) comparison of classification accuracy among origin software, simulation and chip-in-loop emulation.

recovered image is dark with visible noise. As the N_s is larger, the recovered image is more like the origin one, and recovered images with $N_s = 4$ and $N_s = 8$ are almost the same, which is consent with the results in Figure 4(e).

3.3 Array-level boosting for ResNet-34 with chip-in-loop emulation

In DNN based image classification application with memristor based computing-in-memory chips, with the noise-aware training in software, the accuracy of a neural network can tolerate the weight noise when the noise is small. However, when the reading and writing noise of the memristor device is large, significant accuracy loss is still found. Here, we used ResNet-34 to validate the performance of the proposed method. In the mapping process, the weight of each layer is unrolled to a two-dimension matrix and is vertically and horizontally split into several small matrixes to satisfy the size of physical arrays. Then, with the linearly differential mapping method as Section 2, the weight matrix is programmed to the memristor array. As shown in Figure 5(a), when the standard deviation of reading and writing noise is 8% (a comparable amplitude with the noise of fabricated array), the accuracy will drop about 6%. Besides, when the weight precision is less than 5 bits, the accuracy loss is larger than 2%. Thus, it required more techniques to reduce the accuracy loss induced by the noise and limited precision.

Before applying the array-level boosting method with GSEA algorithm to ResNet-34, we need to determine the hyper parameters in the GSEA algorithm. As shown in Figures 5(c) and (d), we show the accuracy loss and allocated arrays of neural network with the spatial extended allocation as the accuracy threshold $\mathrm{Acc_{th}}$ is linearly changed from 0.1% to 1.2% and the standard deviation of noise of the optimized layer is linearly changed from 12% to 40%. In the optimization process of N_s, the size of the memristor array is set to 32×128 to fit the size of the fabricated array, and the maximum spatial extended allocation N_s^m is set to 32 to avoid explosive allocation of memristor array. In Figures 5(c) and (d), when the standard deviation of noise is fixed at about 30% and the accuracy threshold increases alone, the accuracy loss and the number of allocated arrays will slightly increase and rapidly decrease, respectively. It can be explained that a large number of arrays will be allocation to satisfy the requirement of accuracy

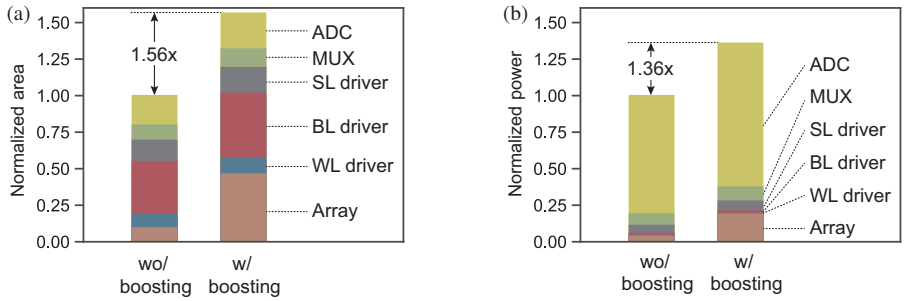

Figure 6 (Color online) Estimation of overhead with array-level boosting allocation. (a) Area usage and its breakdown; (b) power consumption and its breakdown.

threshold when the standard deviation of noise is too large and accuracy threshold is small, whereas the benefit of accuracy recovery is limited, which is illustrated in Figure 4(e). When the accuracy threshold is fixed at 0.6% and the standard deviation of noise increases alone, the accuracy loss rapidly decreases while the number of allocated arrays slightly increases. The reason behind this is similar to the above. To achieve a balance between the accuracy loss and the number of allocated arrays, we choose 0.2% as the accuracy threshold and 28% as the standard deviation of noise. In this condition, the accuracy loss of simulation is 0.06% and the number of allocated arrays is 1289 (about 5 M memristor devices).

To valid the results of the simulation and silicon-based experiment, we developed a framework with a chip-in-loop emulator of memristor based computing-in-memory system (Figure 5(e)). Within the framework, the neural network is first trained with the customized PyTorch based noise-aware training framework to tolerate a certain degree of weight noise. Then, based on the preset accuracy threshold and standard deviation of noise, the GSEA algorithm starts to optimized the N_s of each layer, Next, the weights of the neural network are sliced to satisfy the size of the fabricated array, and these sliced weights and input are scheduled into a chip-in-loop emulator. In the chip-in-loop emulator, sliced weights are programmed sequentially to the 32×128 memristor array and the inference calculation is performed layer by layer. After the programming and computing operations of all layers are finished, the output results will be compared with the labels to generate the chip-in-loop accuracy. The software-based classification accuracy of ResNet-34 on CIFAR-10 dataset is 93.2%. As shown in Figure 5(f), the simulated and chip-in-loop results show that the array-level boosting method can recover the accuracy from 86.1% and 85.6% to 92.6% and 92.3%, respectively, which are closed to the software-based classification accuracy.

In the memristor-based computing-in-memory chips, most of the overhead is induced by the peripheral circuits, such as the driver circuits and analog-to-digital circuits (ADCs). Here, we used an 8-bit ADC implementation [28]. In the breakdown of area usage (Figure 6(a)), the BL driver contributes the largest area due to that the digital-to-analog function is integrated into BL drivers and BLs are four times more than SLs. In the breakdown of power (Figure 6(b)), the ADC dominates the whole power consumption. Hence, to reduce the area and power overhead of the array-level boosting method, the peripheral circuits are shared among multiple arrays, which is similar to the previous work [29]. Here, we adopt XPEsim [30] to estimate the area and power of memristor based computing-in-memory macro. In the simulation, the share number is set to four. The overhead of total area and power with array-level boosting method is 56% and 36% (Figure 6).

4 Conclusion

In summary, the array-level boosting method with spatial extended allocation has been proposed to reduce the accuracy loss of memristor based computing-in-memory chip induced by the reading and writing noise and limited precision. The proposed method can reduce the effects of noise and improve the precision by jointing and averaging the outputs of extended mapped arrays. The performance of array-level boosting method is validated with a fabricated 32×128 memristor array in terms of the image processing and neural network tasks. For the image processing task, a 32×32 DCT on a gray image is demonstrated with the fabricated array where the output error will saturate when the spatial allocation number N_s is larger than four. For image classification tasks with a neural network, the greedy spatial extended allocation algorithm is proposed to optimize the N_s of each layer in a neural network, and ResNet-34 on CIFAR-10 dataset is demonstrated with a chip-in-loop emulator. The results show that

the chip-in-loop accuracy with array-level boosting method is the closed to software-based accuracy with about 56% overhead of area usage and 36% of power consumption. Further, the hyper parameters, accuracy threshold and standard deviation of noise, in greedy spatial extended allocation algorithm are heuristically optimized, thus, a delicate criterion can be explored in future studies to optimize the hyper parameters automatically.

Acknowledgements This work was supported in part by National Key R&D Program of China (Grant No. 2019YFB2205103), National Natural Science Foundation of China (Grant Nos. 92064001, 61851404, 61874169), Beijing Municipal Science and Technology Project (Grant No. Z191100007519008), and Beijing Innovation Center for Future Chips (ICFC).

References

1 LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521: 436–444
2 He K M, Zhang X Y, Ren S Q, et al. Deep residual learning for image recognition. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, 2016. 770–778
3 Devlin J, Chang M W, Lee K, et al. BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019. 4171–4186
4 Lee J, Kim C, Kang S, et al. UNPU: a 50.6TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2018. 218–220
5 Salahuddin S, Ni K, Datta S. The era of hyper-scaling in electronics. Nat Electron, 2018, 1: 442–450
6 Ielmini D, Wong H S P. In-memory computing with resistive switching devices. Nat Electron, 2018, 1: 333–343
7 Zidan M A, Strachan J P, Lu W D. The future of electronics based on memristive systems. Nat Electron, 2018, 1: 22–29
8 Zhang W Q, Gao B, Tang J S, et al. Neuro-inspired computing chips. Nat Electron, 2020, 3: 371–382
9 Biswas A, Chandrakasan A P. Conv-RAM: an energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2018. 488–490
10 Si X, Chen J J, Tu Y N, et al. A twin-8T SRAM computation-in-memory macro for multiple-bit CNN-based machine learning. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2019. 396–397
11 Lu J, Young S, Arel I, et al. A 1 TOPS/W analog deep machine-learning engine with floating-gate storage in 0.13 μm CMOS. IEEE J Solid-State Circ, 2015, 50: 270–281
12 Chen W H, Li K X, Lin W Y, et al. A 65 nm 1 Mb nonvolatile computing-in-memory ReRAM macro with sub-16 ns multiply-and-accumulate for binary DNN AI edge processors. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, 2018. 494–496
13 Mochida R, Kouno K, Hayata Y, et al. A 4M synapses integrated analog ReRAM based 66.5 TOPS/W neural-network processor with cell current controlled writing and flexible network architecture. In: Proceedings of IEEE Symposium on VLSI Technology, Honolulu, 2018. 175–176
14 Nandakumar S R, Le Gallo M, Boybat I, et al. Mixed-precision architecture based on computational memory for training deep neural networks. In: Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), Florence, 2018
15 Kim S, Ishii M, Lewis S, et al. NVM neuromorphic core with 64k-cell (256-by-256) phase change memory synaptic array with on-chip neuron circuits for continuous in-situ learning. In: Proceedings of IEEE International Electron Devices Meeting (IEDM), Washington, 2015
16 Sun X Y, Wang P N, Ni K, et al. Exploiting hybrid precision for training and inference: a 2T-1FeFET based analog synaptic weight cell. In: Proceedings of IEEE International Electron Devices Meeting (IEDM), 2018
17 Prezioso M, Merrikh-Bayat F, Hoskins B D, et al. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. Nature, 2015, 521: 61–64
18 Ambrogio S, Narayanan P, Tsai H, et al. Equivalent-accuracy accelerated neural-network training using analogue memory. Nature, 2018, 558: 60–67
19 Yao P, Wu H Q, Gao B, et al. Face classification using electronic synapses. Nat Commun, 2017, 8: 15199
20 Li C, Wang Z R, Rao M Y, et al. Long short-term memory networks in memristor crossbar arrays. Nat Mach Intell, 2019, 1: 49–57
21 Joshi V, Le Gallo M, Haefeli S, et al. Accurate deep neural network inference using computational phase-change memory. Nat Commun, 2020, 11: 2473
22 Liu B Y, Li H, Chen Y R, et al. Vortex: variation-aware training for memristor X-bar. In: Proceedings of the 52nd ACM/EDAC/IEEE Design Automation Conference, San Francisco, 2015
23 Yao P, Wu H Q, Gao B, et al. Fully hardware-implemented memristor convolutional neural network. Nature, 2020, 577: 641–646
24 Gonugondla S K, Kang M, Shanbhag N R. A variation-tolerant in-memory machine learning classifier via on-chip training. IEEE J Solid-State Circ, 2018, 53: 3163–3173
25 Boybat I, Le Gallo M, Nandakumar S R, et al. Neuromorphic computing with multi-memristive synapses. Nat Commun, 2018, 9: 2514
26 Joksas D, Freitas P, Chai Z, et al. Committee machines — a universal method to deal with non-idealities in memristor-based neural networks. Nat Commun, 2020, 11: 4273
27 Wu W, Wu H Q, Gao B, et al. A methodology to improve linearity of analog RRAM for neuromorphic computing. In: Proceedings of IEEE Symposium on VLSI Technology, Honolulu, 2018. 103–104
28 Kull L, Toifl T, Schmatz M, et al. A 3.1 mW 8b 1.2 GS/s single-channel asynchronous SAR ADC with alternate comparators for enhanced speed in 32 nm digital SOI CMOS. In: Proceedings of IEEE International Solid-State Circuits Conference Digest of Technical Papers, San Francisco, 2013. 468–469
29 Shafiee A, Nag A, Muralimanohar N, et al. ISAAC: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In: Proceedings of the 43rd Annual International Symposium on Computer Architecture (ISCA), Seoul, 2016. 14–26
30 Zhang W Q, Peng X C, Wu H Q, et al. Design guidelines of RRAM based neural-processing-unit: a joint device-circuit-algorithm analysis. In: Proceedings of the 56th Annual Design Automation Conference, Las Vegas, 2019