A High-Speed and High-Efficiency Diverse Error Margin Write-Verify Scheme for an RRAM-Based Neuromorphic Hardware Accelerator

Yudeng Lin, Jianshi Tang, *Senior Member, IEEE*, Bin Gao, *Senior Member, IEEE*, Qi Qin, *Graduate Student Member, IEEE*, Qingtian Zhang, He Qian, and Huaqiang Wu, *Senior Member, IEEE*

***Abstract*—Resistive random access memory (RRAM)-based neuromorphic hardware accelerators are attractive platforms for neural network acceleration due to their high energy efficiency. However, the inherent variations of RRAM, arising from diffusion or recombination of oxygen vacancies, can cause significant conductance deviation from the target value, resulting in noticeable performance degradation. In practical *ex situ* training, write-verify methods are widely adopted to avoid this issue when transferring a trained network model. However, the intense reading and reprogramming operations make the conventional write-verify methods require extensive programming time and energy. In this brief, for the first time, we propose a novel write-verify scheme that can transfer each weight with a different acceptable error margin to achieve a high-speed and high-efficiency write-verify scheme while maintaining network performance. Our experimental results show that the speed and energy efficiency of the write-verify process can be improved significantly, by up to ×3.4∼×9.0 and ×4.1∼×14.1, respectively.**

***Index Terms*—Resistive random access memory (RRAM), neuromorphic accelerator, write-verify, Bayesian method.**

I. Introduction

Resistive random access memory (RRAM) has been extensively studied as a promising candidate for neuromorphic computing [1], [2]. Highly parallel RRAM-based crossbars are attractive platforms for neural network acceleration [3], [4], [5], [6], [7], [8], [9], [10]. In neuromorphic computing, the conductance values of the RRAM cell in crossbars represent the synaptic weights in the network and should be programmed before computation. However, due to the diffusion or recombination of oxygen vacancies in multiple weakly conductive filament regions, the RRAM conductance might fluctuate when programmed [2]. Deviation of programmed weights from the trained target weights caused by variations is inevitable, degrading the network performance significantly.

Two mainstream solutions, *in situ* [3], [4], [5], [6] and *ex situ* training, are proposed to mitigate the impact of conductance fluctuation on network performance. *In situ* training, which directly training on a crossbar array, are effective but require extra complex hardware for backpropagation and weight updating. In contrast, some widely used practical *ex situ* training methods can be easily implemented [7], [8], [9], [10]. The networks are trained using existing software platforms and then transferred to a neuromorphic computing accelerator. To transfer an external trained network model to a crossbar, RRAM cells are programmed to the target conductance states within an accepted error margin by using write-verify operation methods. The write-verify methods can reduce the weight deviation remarkably while maintaining network performance.

However, using an identical error margin for each weight in the write-verify process is extremely energy- and time-consuming since it demands a large amount of reading and programming operations for individual weight. In addition, identical error margins that are too large or small may degrade network performance or reduce transfer efficiency by applying imprecise weights or adding write-verify cycles. It is unacceptable to reprogram large-scale RRAM-based neuromorphic hardware accelerators for different tasks. These issues hinder its application to areas, such as mobile edge computing scenario, which demand high-speed and high-efficiency write-verify schemes for weight transfer. However, an efficient solution for weight transfer is lacking.

A straightforward way to tackle these issues is to relax the limit of the error margin for each weight differently, resulting in fewer reading and programming operations. However, due to the difficulty and complexity of evaluating each weight deviation's influence on network performance, it is still a great challenge to determine every acceptable error margin for weights. In this brief, we present a unified and efficient write-verify scheme using diverse error margins. The major contributions of this brief are summarized below:

- The proposed *ex situ* training method converts the weights in the traditional network into probabilistic distributions

Manuscript received 30 October 2022; accepted 19 November 2022. Date of publication 24 November 2022; date of current version 29 March 2023. This work was supported in part by the Ministry of Science and Technology (MOST) of China under Grant 2021ZD0201200; in part by the National Natural Science Foundation of China under Grant 92064004, Grant 92164302, and Grant 62176163; in part by the XPLORER Prize; and in part by the Beijing Advanced Innovation Center for Integrated Circuits. This brief was recommended by Associate Editor L. A. Camunas-Mesa. *(Corresponding authors: Qingtian Zhang; Huaqiang Wu.)*

Yudeng Lin and Qi Qin are with the School of Integrated Circuits, Beijing Innovation Center for Future Chips (ICFC), Tsinghua University, Beijing 100084, China.

Jianshi Tang, Bin Gao, Qingtian Zhang, He Qian, and Huaqiang Wu are with the School of Integrated Circuits, Beijing Innovation Center for Future Chips and the Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: zhangqt0103@mail.tsinghua.edu.cn; wuhq@tsinghua.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2022.3224470.

Digital Object Identifier 10.1109/TCSII.2022.3224470

without changing the network structure. The proposed method only uses some simple prior parameters of RRAM in a simple training process. Finally, it can obtain optimized trainable parameters, which are related to error margins, in a traditional network.

- A diverse error margin determination method is proposed and can significantly relax the programming accuracy of most RRAM weights differently.
- A diverse error margin write-verify method is proposed; it transfers each weight with a different acceptable error range and can greatly reduce the cost of weight transfer.
- We evaluate the proposed diverse error margin scheme by two typical deep neural networks in a classification task.

II. Preliminaries

A. RRAM-Based Neuromorphic Hardware Accelerator

The basic operation of a deep neural network can be expressed as vector-matrix multiplications (VMMs). A differential conductance pair of RRAM cells finds synaptic weight values in neural networks, so both positive and negative weights can be fully represented by RRAM. The voltages applied on the word line are the input vector of the network layers. The computing result vectors are the accumulated output currents flowing on the bit line. In an RRAM-based VMM operation, we assume that the dimension of the input vector of the network layers and the result vector are n and m, respectively. The entry of the output current vector I_j is:

$$I_j = \sum_{i=1}^{n} V_i \cdot \left(G_{i,j}^{+} - G_{i,j}^{-}\right), j \in (1, 2, \ldots, m)$$

where $W_{i,j} = G_{i,j}^{+} - G_{i,j}^{-}$ is the differential conductance of a pair of RRAM cells, representing one synaptic weight in the network, and V_i is the entry of the input voltages vector.

B. Ex Situ Training

The neural network models are trained to obtain target weights first, utilizing *ex situ* training approaches before hardware computation. In typical *ex situ* training [7], [10], a network training process is performed on a conventional software platform. The weights in the network are optimized until the network achieves the expected reasonable performance. Next, the quantified target conductance values of RRAM corresponding to the weights are obtained. Finally, the target conductance values are transferred into the hardware accelerator. The *ex situ* training method can easily make use of existing high-performing computation platforms. For multiple transfers, the network training process does not need to be repeated since the existing learned weights can be directly transferred into multiple hardware accelerators. Therefore, the *ex situ* training method can be applied on a large scale, especially for edge devices used for inference.

C. Weight Transfer With Verification

The RRAM-based accelerator suffers from various sources of variations and noises [12]. It is difficult to precisely transfer learned weights into the hardware accelerator due to these variations. This kind of transfer error can have a significant impact on a neural network's performance. To address the issue of the transfer error, Alibart et al. [13] proposed a simple, widely used closed-loop feedback scheme to modulate the RRAM conductance. RRAM was programmed first to an initial random conductance state. Then, the conductance value was measured by a read pulse operation to verify whether its conductance is precise. Next, whether its value falls within an identical error margin from the target value was determined. If it failed, RRAM was programmed again with an additional program pulse to ensure that its value falls within the error margin. The conductance value continuously approaches the target value through a series of repeated programs and read operations until it is acceptable.

D. Bayesian Neural Network

Unlike traditional neural networks, where weights are fixed values, Bayesian neural network (BNN) weights are depicted by random variables [14], [15]. A BNN is a parametric model that incorporates the flexibility of neural networks into a Bayesian framework. The learning process of a BNN involves probability distributions. This crucial component incorporates probabilistic weights into the network training process [16]. Thus, the trained weight parameters and calculations must be resilient under tolerable weight deviations. Therefore, a BNN can be used to determine appropriate weights and acceptable diverse error margins. As for the advantages of BNN, it combines the objective function of network tasks with the learning of weight distribution in network training process. What's more, BNN can take the intrinsic non-ideal factors of RRAM, read variation, as the network parameter. The disadvantage of BNN is that it will increase the time consumption of training neural network. However, it can be mitigated by pre-training.

III. Proposed DIVERSE Scheme

In the traditional process of the RRAM-based accelerator, the network training process aims to optimize weights and then transfer the target weights into crossbar arrays with write-verify methods. However, using an identical error margin for each weight in the write-verify process demands notable reading and programming operations, leading to large energy and time consumption. Hence, we propose a high-speed and high-efficiency diverse error margin write-verify (DIVERSE) scheme. The proposed DIVERSE scheme consists of an *ex situ* training method and a write-verify method. The propose training method uses trainable parameters for obtaining acceptable weight deviations. A diverse error margin determination method is proposed to relax the requirement for exact matches between each RRAM weight and target weight. The proposed write-verify method uses different error margins for each weight to reduce the cost of weight transfer significantly.

A. Overview of the DIVERSE Scheme

The DIVERSE scheme involves three major phases, as illustrated in Fig. 1. First, a traditional neural network is transformed into a BNN with the same network structure. The BNN is trained to obtain the appropriate probabilistic weights. The weights are then allocated distinct error margins based on

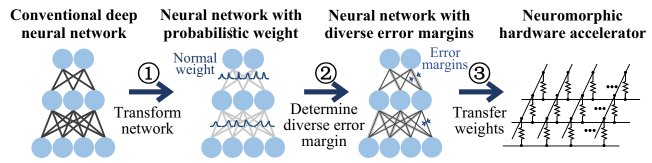
Conventional deep neural network
Neural network with probabilistic weight
Neural network with diverse error margins
Neuromorphic hardware accelerator
① Transform network
Normal weight
② Determine diverse error margin
Error margins
③ Transfer weights

Fig. 1. Basic process of the proposed DIVERSE scheme.

the optimized parameters in the BNN, and the network is eventually transferred onto an RRAM-based accelerator using the proposed diverse error margin write-verify method.

B. Ex Situ Training Using Probabilistic Weights

To acquire acceptable weight deviations without any network performance degradation, a BNN with the same network structure is created from a conventional neural network model. We use Bayes by backprop (BBB) for learning a probability distribution on the parameters in the created BNN, which is an approximate variational method introduced by Blundell et al. [15]. The probabilistic weight in the BNN follows normal distribution $N(\mu_{i,j}, {\sigma_{i,j}}^2)$. And mean μ is identical to the fixed weights in conventional neural network. When using pretrained conventional network models, we can easily set the mean μ to pretrained weight values. The standard deviation σ is additional parameter to capture the uncertainty of weight, which can be optimized easily by stochastic gradient descent [17]. The optimized mean and variance can represent the appropriate deviation of different weights without affecting the network performance, so that different weights can use different write error ranges. In RRAM case, mean $\mu_{i,j}$ is the target weight $\hat{W}_{i,j}$ that is needed to transfer on the crossbar. The deviation of weight $\hat{W}_{i,j}$ is indicated by standard deviation $\sigma_{i,j}$. Training a BNN is main cost of ex situ training step. If we use pretrained network models, the training process is to finetune weights, and the cost can be less than training of a traditional network. If we train the BNN from the beginning without pretrained models, the training cost of a BNN is usually twice of a traditional network with similar architecture, due to the double learnable parameters in a BNN, i.e., mean and variance for every single point estimate weight in the traditional network. However, this training process uses existing software platforms and is one-time cost for the same application. So, in mass hardware accelerators production, it is reasonable to consider the programming cost for multiple transfers.

After training, the optimal probabilistic weights are resilient and ensure network performance. Moreover, there are certain general requirements in the learning process to guarantee that the network is more consistent with the RRAM-based accelerator, such as truncating the weights within an identical symmetric range so that the trained weights can be implemented with differential RRAM cells, and limiting the identical minimum value of standard deviations to ensure the standard deviation is always larger than read variations and the network has better read variation resistance.

Hence, the proposed *ex situ* training employs probabilistic weights in the training process to generate robust target weights and tolerable weight deviations. This ensures that the network's inference output is resistant to change even when the RRAM weights are transferred with various error margins.

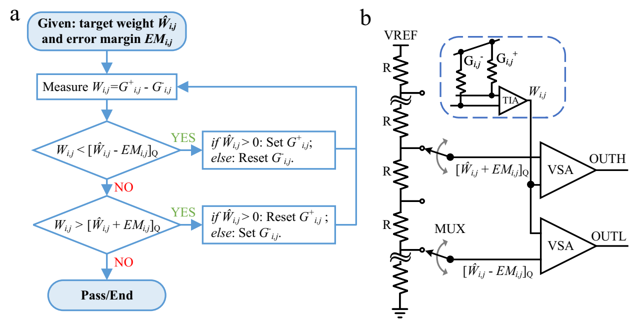
a
Given: target weight $\hat{W}_{i,j}$ and error margin $EM_{i,j}$
Measure $W_{i,j}=G^+_{i,j} - G^-_{i,j}$
$W_{i,j} < [\hat{W}_{i,j} - EM_{i,j}]_Q$
YES
if $\hat{W}_{i,j} > 0$: Set $G^+_{i,j}$; else: Reset $G^-_{i,j}$.
NO
$W_{i,j} > [\hat{W}_{i,j} + EM_{i,j}]_Q$
YES
if $\hat{W}_{i,j} > 0$: Reset $G^+_{i,j}$; else: Set $G^-_{i,j}$.
NO
Pass/End
b
VREF
R
R
R
R
$G_{i,j}^-$
$G_{i,j}^+$
TIA
$W_{i,j}$
$[\hat{W}_{i,j} + EM_{i,j}]_Q$
VSA
OUTH
MUX
$[\hat{W}_{i,j} - EM_{i,j}]_Q$
VSA
OUTL

Fig. 2. (a) Process of the diverse error margin write-verify method. (b) Prototype verify circuit in the DIVERSE scheme.

C. Diverse Error Margin Determination

After learning using the proposed *ex situ* training method, the target weight values in the network are represented by mean μ of the normal distribution. The acceptable deviation of each weight corresponds to standard deviation σ of the normal distribution. The performance of the network is not sensitive to a certain level of deviation (related to σ) of the RRAM weight, which is ensured by the proposed training method. In other words, it is not necessary to transfer weight using a small identical error margin. The larger standard deviation σ is, the larger the acceptable deviation of the weight. Hence, we assign different error margins for the weights, which are proportional to standard deviation σ. We can formulate the error margin as:

$$EM_{i,j} = k \cdot \sigma_{i,j}$$

where $EM_{i,j}$ and $\sigma_{i,j}$ are the error margin and learned standard deviation of weight $\hat{W}_{i,j}$ in the network, respectively. k is the proportionality factor, which is the same for every weight and is determined by the network and learning task. Then, the upper and lower tolerance boundaries of weight $\hat{W}_{i,j}$ are:

$$\begin{cases} BU_{i,j} = \left[\hat{W}_{i,j} - EM_{i,j}\right]_Q \\ BL_{i,j} = \left[\hat{W}_{i,j} + EM_{i,j}\right]_Q \end{cases},$$

where $[\,\cdot\,]_Q$ is the quantization operation. Therefore, we can use different error margins to transfer the weights with a write-verify scheme to relax the requirement for the circuit.

D. Transfer With Diverse Error Margin Write-Verification

To transfer the weight in a network into RRAM conductance, a diverse error margin write-verify method is proposed, as illustrated in Fig. 2a. The target weight $\hat{W}_{i,j}$ and the corresponding acceptable error margin $EM_{i,j}$ can be determined after completing the previous two phases. The proposed method measures the present RRAM weights of the differential pair $W_{i,j} = G^+_{i,j} - G^-_{i,j}$. The verification process checks whether the deviation is within the allowable error margin by comparing the present weight value $W_{i,j}$ with tolerance boundaries $BU_{i,j}$ and $BL_{i,j}$. If the verification result is "pass", the write verification procedure is completed. Otherwise, the conductance of RRAM is programmed to minimize the deviation value until the verify result is "pass". Fig. 2b shows a prototype verification circuit, and the verify logic is the same as Fig.4a for tightening conductance distributions. Each verified

VSA is connected to a reference voltage that represents a conductance threshold. The reference voltages are generated by a resistor voltage divider network. The circuit aims to tighten RRAM conductance within the window formed by the high threshold and low threshold ($BU_{i,j}$ and $BL_{i,j}$). As the conventional write-verify method, all RRAM devices in the same column share a common verification circuit, however, do not share common thresholds. In fact, when verifying each RRAM device, reconfiguring the $BU_{i,j}$ and $BL_{i,j}$ is required for the conventional method as well as our method.

IV. Experimental Evaluation

A. Experimental Setup

Two common deep neural networks, a multilayer perceptron (MLP) and a CNN called LeNet, are utilized to validate our proposed scheme on MNIST dataset. First, the networks are transformed into BNNs with the same network structure, and the BNNs are trained on the MNIST training set using the proposed *ex situ* training method. We employ the widely-used Adam algorithm for training. The conductance window ratio of RRAM is 10.0, and the minimum conductance is 2.0 μS. Then, the tolerance boundaries are quantified to n conductance levels.

Next, we transfer weights to the RRAM-based accelerator through the DIVERSE scheme. We obtain the statistics of conductance variation from [18] to establish the simulated RRAM program variation model in the evaluated experiments. The analog switching data are measured using identical pulses during the program process. The program variation model is: $G' = G + Log\_Normal(\Delta\ G_{\text{ideal}}, S)$, where the ideal conductance change ($\Delta\ G_{\text{ideal}}$) and the update variations factor (S), which depend on the current conductance state (G) and operation direction (SET or RESET), are obtained from [21]. The read variation model is: $G_{read}' = G' + Normal(0, \sigma^2{}_{read})$, which is used in evaluating the final classification drop under different read variations σ_{read}, is a simple additive normal noise model. We get the minimum value of the standard deviation through presented data measured on a physical RRAM-based accelerator in [19], which is small conductance fluctuation mainly originating from RTN and σ_{read} = 0.2 μS. For the application of computing in memory, the endurance of our RRAM device can reach 1e6 [20] and retention time is 1e4 s @ 85°C [21]. It is found that there is almost no influence for endurance and retention due to the robust tolerance of neuromorphic computing system. So, the endurance and retention are not considered in our experiments. However, we should point out that the proposed method can save writing-verify iterations and programming time, which further improve the device endurance and retention.

Finally, using the MNIST test set, the accuracy drop of the network is evaluated by dividing the number of incorrect predictions made by the total number of predictions made.

The network's accuracy drop is stochastic due to random weight variations. We perform weight transfer 20 times to carefully analyze the resulting average accuracy drop in each scenario. We study the impact of the DIVERSE scheme by testing various scenarios of the conductance level n and the proportionality factor k. A combination of one program and one read operation is regarded as a single write-verify cycle. The time spent on weight transfer is proportional to the write-verify cycle number. Therefore, we average write-verify

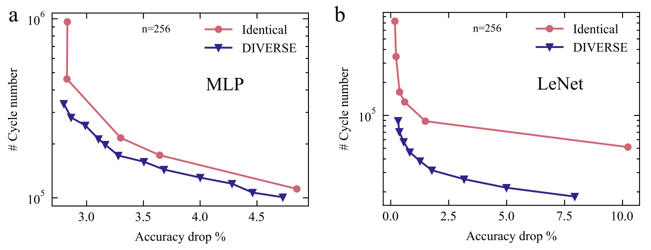

Fig. 3. Required write-verify cycle number varies with the network accuracy drop (a) MLP and (b) LeNet.

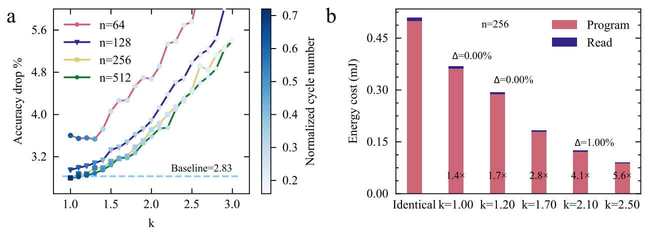

Fig. 4. (a) Accuracy drop and normalized write-verify cycle for MLP with different conductance levels n and proportionality factors k of error margins. (b) Energy cost in the identical error margin case and different error margin weight transfer cases.

cycle number for 20 weight transfers to measure the time consumed. The energy cost is evaluated by accumulating the energy consumption required for each program or read pulse operation. For program operation, the set and reset voltages are 1.3 V and 2.0 V, respectively. The read voltage is 0.2 V. The voltage pulse width is 50 ns for both the program and read operations.

B. Experiments on MLP

A two-layer MLP with 100 neurons in the hidden layer is used to validate the effectiveness of the DIVERSE scheme, which has 0.15M weights and is represented by 0.30M RRAM cells. The conventional weight transfer method employs identical error margins (IEM) to transfer whole weights. The accuracy drops more and the write-verify cycle is required less as the IEM value increases, as shown in Fig. 3a. When IEM is 0.56 µS, it keeps a small accuracy drop of around 2.83%, and it requires 4.61×10^5 write-verify cycles, which is less than other smaller identical error margin cases. Fig. 3a shows that the DIVERSE scheme requires fewer write-verify cycles than the conventional method with the same accuracy drop.

As shown in Fig. 4a, we study the impact of different error margin proportionality factors k on the number of write-verify cycles and the accuracy drop of the RRAM network. The solid lines with markers indicate the accuracy drop under different conductance levels n. In addition, the color of the marker is used to indicate the normalized write-verify cycle number related to the conventional method. When n is equal to 256, the complexity of the verify circuit and the performance of the network are balanced, as shown in Fig. 4a. In this scenario, the required cycle number for the DIVERSE scheme is reduced to 3.35×10^5 when $k = 1.00$. The relative accuracy loss Δ is zero ($\Delta = 0.00\%$) even when the error margin is $k = 1.20$, which is almost less than 1.88×10^5 write-verify cycles, and the energy efficiency is improved by $\times1.7$ compared with the conventional weight transfer method (Fig. 4b). When $k = 2.10$, the cycle is further reduced to 29.59% ($\times3.4$), and the energy

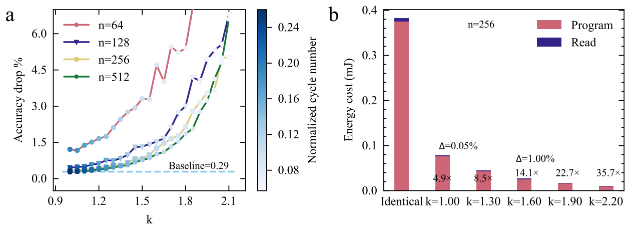

Fig. 5. (a) Accuracy drop and normalized write-verify cycle for LeNet with different conductance levels n and different error margin factors k. (b) Energy cost in the identical error margin case and different error margin weight transfer cases.

efficiency is improved by $\times 4.1$ even though the accuracy loss Δ is 1.00%. These results indicate that reasonable diverse error margin settings can significantly reduce the number of write cycles to save time and energy while maintaining accuracy.

C. Experiments on LeNet

The proposed DIVERSE scheme is tested using LeNet, a typical network with convolution layers. The network is made up of 2 convolution layers, 2 max pooling layers, and 3 fully connected layers, which has 0.12M weights and is represented by 0.24M RRAM cells. In comparison to the previous MLP model, it is a fairly deep neural network. After being trained and transferred with the traditional weight transfer method, the final accuracy drop of LeNet is 0.29%, and 3.45×10^5 write cycles are needed, as shown in Fig. 3b. The figure also shows that the DIVERSE scheme requires fewer write-verify cycles than the conventional method with the same accuracy drop. The same experiments as those for the MLP model are carried out, and the results are presented in Fig. 5.

When the conductance level n equals 256, a better trade-off between verification circuit complexity and network performance is obtained, as shown in Fig. 5a. In such scenario (n=256 and $k = 1.00$), the weight transfer cycle is reduced to 0.89×10^5, and energy efficiency is improved by $\times 4.9$ with a slight accuracy loss ($\Delta = 0.05\%$). Moreover, the accuracy remains almost unchanged ($\Delta = 1.00\%$) when $k = 1.60$. It improves the energy efficiency almost 14.1 times compared to the traditional method, as shown in Fig. 5b. These experiments show that the DIVERSE scheme only requires 3.83×10^4 write cycles ($\times 9.0$), leading to high-speed and high-efficiency weight transfer. Also, Fig. 4a and Fig. 5a show that LeNet is more sensitive to error margins. Its accuracy drops faster than MLP's accuracy as proportionality factor k increases. This might be caused by the large difference in the weights of convolution layers, which will impact the capability to extract features.

V. Conclusion

This brief proposed a DIVERSE scheme to achieve high-speed and high-efficiency weight transfer for RRAM-based accelerators. The scheme acquires a different tolerable error margin, which can greatly relax the constraint for weight deviation from the target value. The experimental results reveal that the DIVERSE scheme can significantly improve efficiency by $\times 3.4\sim\times 9.0$ and $\times 4.1\sim\times 14.1$ in speed and energy, respectively.

References

[1] M. Lanza et al., "Memristive technologies for data storage, computation, encryption, and radio-frequency communication," *Science*, vol. 376, no. 6597, p. eabj9979, 2022, doi: 10.1126/science.abj9979.

[2] C. Kaspar, B. J. Ravoo, W. G. van der Wiel, S. V. Wegner, and W. H. P. Pernice, "The rise of intelligent matter," *Nature*, vol. 594, no. 7863, pp. 345–355, 2021, doi: 10.1038/s41586-021-03453-y.

[3] P.-Y. Chen, L. Gao, and S. Yu, "Design of resistive synaptic array for implementing on-chip sparse learning," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 2, no. 4, pp. 257–264, Oct.–Dec. 2016, doi: 10.1109/TMSCS.2016.2598742.

[4] C. Li et al., "Efficient and self-adaptive in-situ learning in multilayer memristor neural networks," *Nat. Commun.*, vol. 9, no. 1, p. 2385, 2018, doi: 10.1038/s41467-018-04484-2.

[5] F. Alibart, E. Zamanidoost, and D. B. Strukov, "Pattern classification by memristive crossbar circuits using ex situ and in situ training," *Nat. Commun.*, vol. 4, pp. 1–7, Jun. 2013, doi: 10.1038/ncomms3072.

[6] P. Yao et al., "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, pp. 641–646, Jan. 2020.

[7] M. Hu et al., "Memristor-based analog computation and neural network classification with a dot product engine," *Adv. Mater.*, vol. 30, no. 9, 2018, Art. no. 1705914, doi: 10.1002/adma.201705914.

[8] Y. Lin et al., "Bayesian neural network realization by exploiting inherent stochastic characteristics of analog RRAM," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, 2019, pp. 1–4, doi: 10.1109/IEDM19573.2019.8993616.

[9] M. Hu, J. P. Strachan, Z. Li, and R. S. Williams, "Dot-product engine as computing memory to accelerate machine learning algorithms," in *Proc. 17th Int. Symp. Qual. Electron. Des. (ISQED)*, 2016, pp. 374–379, doi: 10.1109/ISQED.2016.7479230.

[10] P. Lin et al., "Three-dimensional memristor circuits as complex neural networks," *Nat. Electron.*, vol. 3, no. 4, pp. 225–232, 2020, doi: 10.1038/s41928-020-0397-9.

[11] D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," *Nat. Electron.*, vol. 1, no. 6, pp. 333–343, 2018, doi: 10.1038/s41928-018-0092-2.

[12] V. Yon, A. Amirsoleimani, F. Alibart, R. G. Melko, D. Drouin, and Y. Beilliard, "Exploiting non-idealities of resistive switching memories for efficient machine learning," *Front. Electron.*, vol. 3, pp. 1–11, Mar. 2022, doi: 10.3389/felec.2022.825077.

[13] F. Alibart, L. Gao, B. D. Hoskins, and D. B. Strukov, "High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm," *Nanotechnology*, vol. 23, no. 7, 2012, Art. no. 75201, doi: 10.1088/0957-4484/23/7/075201.

[14] H. Wang and D.-Y. Yeung, "A survey on Bayesian deep learning," 2016, *arXiv:1604.01662*.

[15] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, vol. 2, 2015, pp. 1613–1622.

[16] A. Foong, D. Burt, Y. Li, and R. Turner, "On the expressiveness of approximate inference in Bayesian neural networks," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 15897–15908.

[17] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*, 2nd ed., G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Berlin, Germany: Springer, 2012, pp. 421–436, doi: 10.1007/978-3-642-35289-8_25.

[18] Y. Lin et al., "Demonstration of generative adversarial network by intrinsic random noises of analog RRAM devices,'‘in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, 2018, pp. 67–70, doi: 10.1109/IEDM.2018.8614483.

[19] Q. Hu et al., "Identifying relaxation and random telegraph noises in filamentary analog RRAM for neuromorphic computing," in *Proc. 5th IEEE Electron Devices Technol. Manuf. Conf. (EDTM)*, 2021, pp. 6–8, doi: 10.1109/EDTM50988.2021.9420888.

[20] M. Zhao et al., "Characterizing endurance degradation of incremental switching in analog RRAM for neuromorphic systems," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, San Francisco, CA, USA, Dec. 2018, pp. 1–4, doi: 10.1109/IEDM.2018.8614664.

[21] M. Zhao et al., "Crossbar-level retention characterization in analog RRAM array-based computation-in-memory system," *IEEE Trans. Electron Devices*, vol. 68, no. 8, pp. 3813–3818, Aug. 2021, doi: 10.1109/TED.2021.3089561.