SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

February 2025, Vol. 68, Iss. 2, 122403:1–122403:15
https://doi.org/10.1007/s11432-023-4021-y

Mitigating methodology of hardware non-ideal characteristics for non-volatile memory based neural networks

Lixia HAN[1,2], Peng HUANG[1,2*], Yijiao WANG[3*], Zheng ZHOU[1,2], Haozhang YANG[1,2], Yiyang CHEN[1,2], Xiaoyan LIU[1,2] & Jinfeng KANG[1,2]

[1]*School of Integrated Circuits, Peking University, Beijing 100871, China*
[2]*Beijing Advanced Innovation Center for Integrated Circuits, Beijing 100871, China*
[3]*School of Integrated Circuit Science and Engineering, Beihang University, Beijing 100191, China*

Received 20 August 2023/Revised 26 December 2023/Accepted 29 April 2024/Published online 13 January 2025

Abstract Non-volatile memory-based computing-in-memory (nvCIM) paradigm has been extensively studied to boost the energy efficiency of neural network accelerators in edge applications. However, the degradation of inference accuracy induced by the non-ideal characteristics across circuits, arrays, and devices is becoming a crucial issue. In this work, we establish a hardware characteristic behavior model to analyze the impact of nvCIM non-ideal characteristics on neural network accuracy. Then we propose a hardware aware training and weight mapping correction methods to mitigate inference accuracy degradation. Through simulation verification, about 95% inference accuracy degradation is recovered by adopting the proposed mitigation method for various non-ideal characteristics and various neural network models. The feasibility of the proposed method is further proved in an experimental example with a flash-based LeNet recognition system.

Keywords computing in memory, neural network accelerators, non-volatile memory, hardware non-ideal characteristics, software-hardware co-design

Citation Han L X, Huang P, Wang Y J, et al. Mitigating methodology of hardware non-ideal characteristics for non-volatile memory based neural networks. Sci China Inf Sci, 2025, 68(2): 122403, https://doi.org/10.1007/s11432-023-4021-y

1 Introduction

Deep neural networks (DNNs) have achieved beyond human-level accuracy in pattern recognition and natural language processing, but conventional DNN processors (CPUs, GPUs, FPGAs) encounter great challenges from computation throughput and energy efficiency due to slowing down Moore's Law and von Neumann's bottleneck. Non-volatile memory-based computing-in-memory (nvCIM) is extensively studied as a potential computing paradigm to improve the energy efficiency and throughput of processors. Two-terminal non-volatile memory (NVM) devices, such as resistive random access memory (RRAM) [1–5], phase change memory (PCM) [6, 7], magnetoresistive random access memory (MRAM) [8, 9] and three-terminal NVM devices such as Flash [10–13], FeFET [14, 15] all are explored widely as storage and computation media. Due to vector-matrix multiplication (VMM) can be in-situ implemented in the NVM crossbar with high parallelism, the nvCIM-based DNN accelerators are regarded as a promising route of DNN hardware accelerators. Several nvCIM macros and chips have been demonstrated with more than 10 TOPS/W energy efficiency and 100 GOPS computing throughput [3, 4, 6–8].

However, the non-ideal characteristics of nvCIM-based accelerators would lead to significant degradation of inference accuracy. In edge applications, the DNN training process is generally executed in conventional digital processors rather than nvCIM accelerators to avoid the enormous training cost. Different from conventional digital processors, the nvCIM accelerators implement the VMM computation by analog and mixed-signal circuits. In nvCIM-based DNN accelerators, the weights are converted into conductance and stored in the NVM crossbar in advance. The images are converted into input voltage and then applied to the NVM crossbar. The VMM computation is in-situ implemented according to Ohm's law and Kirchhoff's current law. The VMM results are obtained by reading the output current and finally converted to digital signals by peripheral circuits for post-processing and storage.

* Corresponding author (email: phwang@pku.edu.cn, yijiaowang@buaa.edu.cn)

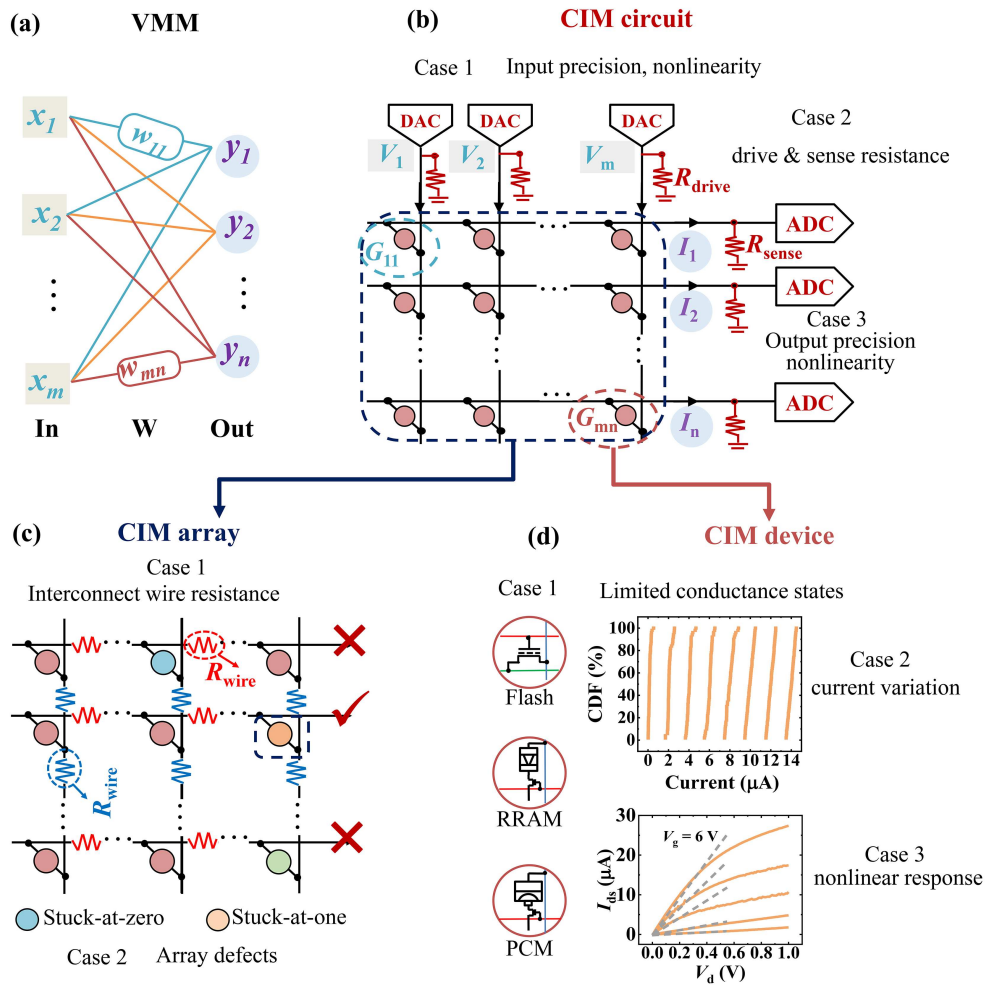

Figure 1 (Color online) nvCIM-based DNNs suffer from nonidealities across circuits, arrays, and devices. (a) The VMM; (b) circuit-level nonidealities include limited input/output precision, nonlinearity, and parasitic resistance; (c) array-level nonidealities include IR drop and stuck-at faults; (d) device-level nonidealities include limited conductance states, device variation, and nonlinear I-V response.

The non-ideal characteristics of nvCIM accelerators originate from analog devices, arrays, and mixed-signal circuits. (1) The nonidealities at the circuit level are mainly from Case 1: limited input coding precision and input coding nonlinearity; Case 2: the parasitic resistances of the driving circuit (R_d) and the sensing circuit (R_s); Case 3: limited output coding precision and output coding nonlinearity, as shown in Figure 1(b). (2) The main nonidealities at the array level result from Case 1: interconnect resistance R_{wire} [16]; Case 2: array defects caused by stuck-at-zero faults (SAF0) and stuck-at-one faults (SAF1) [17], as shown in Figure 1(c). (3) The nonidealities at the device level originate from Case 1: weight quantization errors due to limited NVM conductance states [10, 18]; Case 2: current variation [17, 19, 20], such as programming error and random telegraph noise (RTN); Case 3: the nonlinear I-V response of NVM device [21, 22], as shown in Figure 1(d). All of these deviations between software and nvCIM accelerators would cause the VMM computation error and degrade the inference accuracy of the nvCIM-based DNN accelerators.

Several studies [23–27] have proposed open frameworks for evaluating the impact of CIM hardware characteristics on DNN inference accuracy. Research articles in [23–26] investigated the impact of quantization, nonlinearity, variation, on the training/inference accuracy by integrating compact models. Studies presented in [27] evaluated the DNN inference accuracy with hardware nonidealities by training prediction models using extensive HSPICE simulation results. Refs. [23, 25] targeted on-chip training chips, but a significant amount of simulation time is consumed to evaluate the impact of nonidealities on weight updates. Refs. [24, 27] were dedicated to recovering inference accuracy through off-chip re-training. In addition, various approaches have also been proposed to alleviate the impacts of single-level nonidealities,

Table 1 Comparison between representative studies and this work about the impact of nvCIM nonidealities on DNN accuracy.

	[23]	[27]	[24]	[25]	[26]	This work
Quantization	✓	✓	✓	✓	✓	✓
Variation	✓	✓	✓	✓	✓	✓
Array defect			✓	✓	✓	✓
Nonlinearity	✓	✓		✓	✓	✓
IR drop				✓	✓	✓
Mitigating Solutions	–	–	Hardware aware retraining	Hardware aware retraining	Hardware aware retraining	Hardware aware retraining WMC
Compute Speed	Slow	Slow	Fast	Medium	Medium	Fast
Hardware validation	No	No	No	No	No	Yes

including hardware solutions for array defects [28], device drift [29], device variation [30,31]; and software solutions for device-level characteristics [17, 19, 32], array-level IR drop [16, 33, 34]. To the best of our knowledge, there is still an absence of a thoroughly analyzed and experimentally validated methodology to mitigate the degradation of inference accuracy of nvCIM accelerators.

Table 1 [23–27] summarizes the comparison results between representative studies and this work about mitigating the impact of hardware non-ideal characteristics on neural network accuracy. This work is distinguished by three main aspects: exploring broader nonidealities, proposing richer mitigating methods, and conducting more comprehensive experiments. (1) This work mitigates a broader range of non-ideal factors, including device, array, and circuit-level nonidealities. Based on their mathematical representations, this work categorizes the nonidealities into quantization, variation, defects, nonlinearity, and IR drop. (2) This work proposes two mitigation approaches for nonidealities with different solving complexity to reduce mitigating overheads, utilizing hardware aware retraining for those with lower complexity and weight mapping correction (WMC) for those with higher complexity. (3) This work conducts experiments on a flash-based nvCIM chip to demonstrate the effectiveness of the proposed mitigating methods. Besides the simulation verification, the experiments also prove the feasibility of the proposed mitigating method.

This article is organized as follows: in Section 2, the nvCIM-based VMM computation characteristic behavior model is established; in Section 3, mitigation methods of hardware non-ideal characteristics are proposed, including hardware aware training and weights mapping correction; in Section 4, simulation results of the proposed method are presented; in Section 5, flash-based hardware verification is conducted; in Section 6, the conclusion is drawn.

2 NvCIM hardware characteristics analysis

According to the mathematical representation of non-ideal characteristics, we divide the non-ideal characteristics of nvCIM accelerators into five categories, as shown in Figure 2. The first category is quantization error caused by limited input/weight/output coding precision (such as limited ADC/DAC resolution and NVM conductance states). The second category is weight variation errors (such as programming errors, RTN noise). The third category is weight stationary defects caused by stuck-at-zero and stuck-at-one faults. The fourth category is the nonlinear error caused by input/output nonlinear coding and NVM nonlinear I-V response. The last category is IR drop error caused by interconnect resistance, driving resistance and sensing resistance. All nonidealities parameters are summarized in Table 2.

The ideal VMM computation executed by nvCIM accelerators can be expressed as (1). G_{ij} is the NVM conductance, V_i is the applied voltage and I_j is the output current.

$$I_j = \sum_{i=1}^{m} V_i \cdot G_{ij}. \tag{1}$$

Quantization. Generally, nvCIM accelerators adopt fixed-point precision to implement VMM computation. Therefore, the input, weight and partial sums of DNN models are required to quantize before deploying onto nvCIM accelerators, as shown in Figure 2(a). Considering the ω-bit quantization precision of input $Q_{\text{in}}(V)$, weight $Q_{\text{nvm}}(G)$ and partial sums $Q_{\text{out}}(I)$, the VMM computation performed by nvCIM

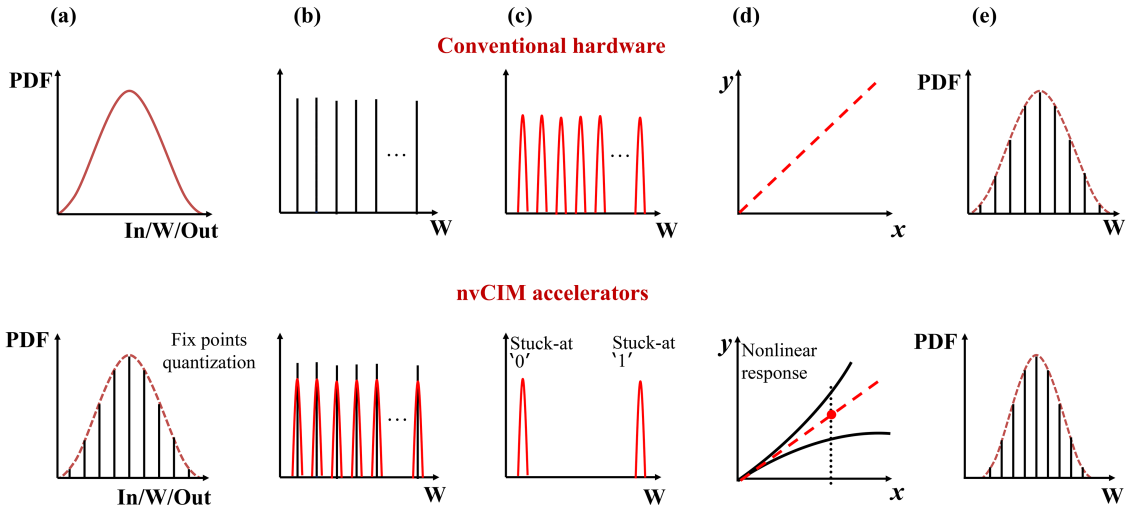

Figure 2 (Color online) Five non-ideal characteristics of nvCIM accelerators. (a) Quantization; (b) variation; (c) array defect; (d) nonlinear response; (e) IR drop.

Table 2 Summaries of nvCIM hardware nonidealities and corresponding parameters.

	nvCIM hardware	Parameters
Quantization	Input precision	$Q_{\text{in}}(V), \omega_{\text{in}}$
	Weight precision	$Q_{\text{nvm}}(G), \omega_w$
	Output precision	$Q_{\text{out}}(I), \omega_{\text{out}}$
Varioation	Weight variation	$f_w(G, \sigma_w), \omega_w, \sigma, N$
Array defect	Stuck at faults	$g(G, G_{\text{saf0}}, G_{\text{saf1}})$
Nonlinearity	Input nonlinearity	$f_{\text{in}}(V)$
	I-V nonlinearity	$f_{\text{iv}}(V, G)$
	Output nonlinearity	$f_{\text{out}}(I)$
IR drop	Interconnect resistance	R_{wire}
	Drive/sense resistance	R_d, R_s

accelerators should be rewritten as

$$I_j = \sum_{k=1}^{m/l} Q_{\text{out}} \left(\sum_{i=1}^{l} Q_{\text{in}}(V_i) \right) \cdot Q_{\text{nvm}}(G_{ij}), \tag{2}$$

where l is the parallelism per VMM computation and m/l is the number of partial sums.

Variation. NVM conductance variation results from programming errors and various noises. The statistical programmed conductance presents a distribution around the target conductance, as shown in Figure 2(b). Here, we assume that the standard deviation of device conductance variation is σ, and the maximum conductance and minimum conductance are G_{max} and G_{min}. Provided that the weight precision is ω_w bit and the single device precision is N bits, ω_w/N devices are required to represent the single weight. Therefore, the standard deviation of weight variation σ_w can be expressed as

$$\sigma_w = \sqrt{\frac{(2^{\omega_w} - 1) \cdot (2^N - 1)}{2^N + 1}} \cdot \frac{\sigma}{G_{\text{max}} - G_{\text{min}}}. \tag{3}$$

The variation standard deviation (σ) of the NVM device can be derived from measured data [10,35]. Taking weight variation f_w into account, the actual weights in nvCIM accelerators should be rewritten as follows:

$$G_{\text{act}} = f_w \left(Q_{\text{nvm}}(G_{ij}), \sigma_w \right). \tag{4}$$

Array defect. Some devices are stuck at low resistance (SAF1) or high resistance (SAF0) during the fabrication process or switching operation. The weights mapped to SAF0 and SAF1 devices only can represent the minimum and maximum weights, as shown in Figure 2(c). The defect distribution function $g(G, G_{\text{saf0}}, G_{\text{saf1}})$ (including defect type and exact defect position) can be detected by using the enhanced

March RC algorithm [36] and squeeze search algorithm [37]. Considering these array defects, the actual weights in nvCIM accelerators should be rewritten as

$$G_{\text{act}} = g\left(f_w\left(Q_{\text{nvm}}(G_{ij}), \sigma_w\right), G_{\text{saf0}}, G_{\text{saf1}}\right). \tag{5}$$

Nonlinearity. Input/output coding circuit and NVM device generally show nonlinear response, as shown in Figure 2(d). We define the nonlinearity NL as the ratio of the absolute response difference integral value to the ideal linear response integral value over the excitation range (x_0–x_1). The response difference is between the ideal linear response Y_l and the actual nonlinear response Y_{nl}. The nonlinearity NL can be expressed as

$$\text{NL} = \frac{\int_{x_0}^{x_1} |Y_l - Y_{\text{nl}}| \, \mathrm{d}x}{\int_{x_0}^{x_1} |Y_l| \, \mathrm{d}x}. \tag{6}$$

Considering the nonlinearity of input coding f_{in}, I-V response f_{iv} and output coding f_{out}, the VMM computation performed by nvCIM accelerators should be rewritten as

$$\begin{aligned} I_j^k &= \sum_{i=1}^{l} f_{\text{iv}}\left(f_{\text{in}}\left(Q_{\text{in}}(V_i)\right), G_{\text{act}}\right), \\ I_j &= \sum_{k=1}^{m/l} f_{\text{out}}\left(Q_{\text{out}}(I_j^k)\right), \end{aligned} \tag{7}$$

where I_j^k is the current of VMM computation partial sums.

IR drop. Considering the wire resistance R_{wire}, driving resistance R_d, and sensing resistance R_s in series, the actual NVM conductance would deviate from the target conductance, especially scaling down to advanced technology, as shown in Figure 2(e). Evaluation of IR drop requires solving the Kirchhoff current equations of the resistor network. Whether solving numerically or iteratively, the evaluation process is time-consuming in the case of large crossbars [38]. Considering IR drop function h, VMM computation partial sums in (7) performed by nvCIM accelerators should be rewritten as

$$I_j^k = \sum_{i=1}^{l} h\left(f_{\text{iv}}\left(f_{\text{in}}\left(Q_{\text{in}}(V_i)\right), G_{\text{act}}\right), R_{\text{wire}}, R_s, R_d\right). \tag{8}$$

We analyze the impact of the above nvCIM nonidealities on CIFAR10 inference accuracy with the VGG11 model. We take the following nvCIM parameters into account, including 8-bit input, 4-bit weight and 8-bit output quantization, 2% $\sigma(G)/(G_{\max} - G_{\min})$ variation, 1% SAF0 and 0.02% SAF1, 2 Ω interconnect resistance R_{wire} (NVM array with 10^5 Ω R_{on} and 10^7 Ω R_{off}), and 8.7% NVM I-V nonlinearity. Figure 3 shows that all the above nonidealities would degrade the inference accuracy seriously. Moreover, nvCIM accelerators usually are exposed to multiple nonidealities. The superposition of these nonidealities would further aggravate the decline of inference accuracy. Therefore, mitigating methods to recover inference accuracy with subtle hardware and software overhead are highly desired.

3 Mitigating method of hardware non-ideal characteristics

3.1 Overview

To decrease the DNN accuracy degradation induced by the above nvCIM nonidealities, we propose mitigating methods for non-volatile memory based neural networks. The flow diagram of the proposed mitigation method is shown in Figure 4.

Before adopting the mitigation methods, nvCIM characteristics and corresponding parameters are required to be measured and extracted. Adopting the mitigation methods, the above nvCIM characteristics with low computational complexity and high computational complexity are injected into the hardware aware training module and the WMC module respectively to address the inference accuracy degradation. In the execution flow, the DNN model is firstly pre-trained to reduce the training epochs and increase the convergence speed of subsequent hardware aware training. Then, hardware aware training method is adopted to update DNN weights to enhance the adaptability of the algorithm to nvCIM characteristics

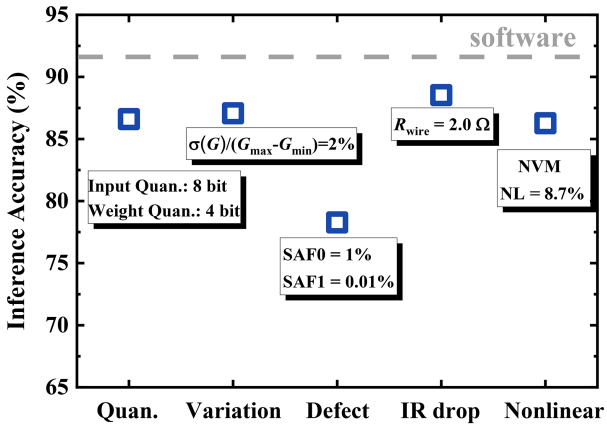

Figure 3 (Color online) Inference accuracy of the VGG11 to recognize CIFAR10 considering circuit, array and device nonidealities.

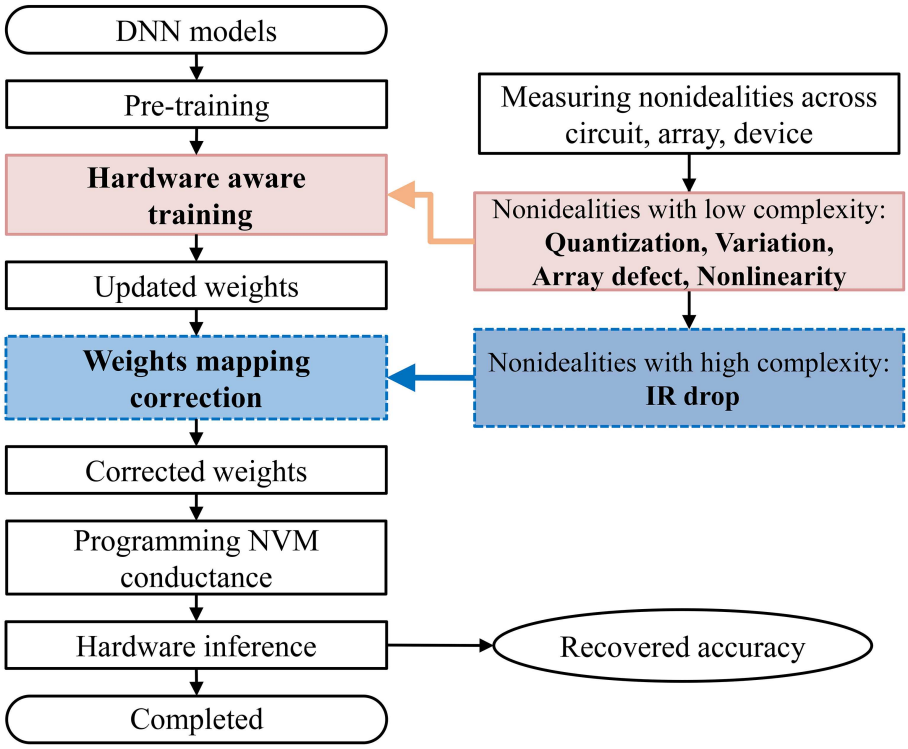

Figure 4 (Color online) Flow diagram of the proposed mitigation method for nvCIM-based DNN accelerators.

by injecting quantization, variation, defect, and nonlinearity into the training process. After that, the WMC method is adopted to correct DNN weights to compensate for the weight deviations induced by IR drop. Finally, the corrected weights are programmed to the NVM conductance and hardware inference is performed to obtain recovered accuracy.

3.2 Hardware aware training

The principle of hardware aware training (HWA) is to inject nvCIM hardware characteristics with low computational complexity into the DNN training process so that the DNN models can actively adapt the nvCIM nonidealities to reduce the inference accuracy degradation. Before hardware aware training, quantization precision, device conductance distribution, array defect type and position, and nonlinear response are required to measure and extract.

The VMM in the DNN algorithm is calculated as follows:

$$Y_j = \sum_{i=1}^{m} X_i \cdot W_{ij}. \tag{9}$$

During the forward propagation of HWA training, the weight matrix (W) is split into positive weights (W^p) and negative weights (W^n). Then, the positive and negative weights are converted into conductance according to the proportional coefficient K_G.

$$G_{ij}^p = K_G \cdot W_{ij}^p \quad \text{and} \quad G_{ij}^n = K_G \cdot W_{ij}^n. \tag{10}$$

Taking limited NVM quantization precision Q_{nvm}, device variation $f_w(G,\ \sigma_w)$ and the SAF array defect distribution function $g(G, G_{\text{saf0}}, G_{\text{saf1}})$ into account, the actual positive and negative weights can be expressed as

$$\begin{aligned} G_{ij}^p &= g\left(f_w\left(Q_{\text{nvm}}\left(G_{ij}^p\right), \sigma_w\right), G_{\text{saf0}}, G_{\text{saf1}}\right), \\ G_{ij}^n &= g\left(f_w\left(Q_{\text{nvm}}\left(G_{ij}^n\right), \sigma_w\right), G_{\text{saf0}}, G_{\text{saf1}}\right). \end{aligned} \tag{11}$$

The input data (X) is converted into input voltage (V) according to the proportional coefficient K_V.

$$V_i = K_V \cdot X_i. \tag{12}$$

We adopt polynomials to fit NVM nonlinear I-V response f_{iv} for multilevel NVM conductance G. We fit N curves $f_{\text{NL}}(V)$ for different NVM conductance G (N bits per cell). We convert I-V nonlinear response into nonlinear voltage $f_{\text{NL}}(V)$ applied to linear conductance.

$$I = f_{\text{iv}}(V, G) = (k_1 V + k_2 V^2 + k_3 V^3 + \cdots) \cdot G = f_{\text{NL}}(V) \cdot G. \tag{13}$$

Taking limited input coding precision, input coding nonlinearity and NVM nonlinear I-V response into account, the actual input voltage can be expressed as

$$V_i = f_{\text{NL}}\left(f_{\text{in}}\left(Q_{\text{in}}\left(V_i\right)\right)\right). \tag{14}$$

After applying input voltage in (14) to the NVM conductance in (11), positive and negative output currents are subtracted and quantified by output coding circuits. Considering limited output coding precision, output coding nonlinearity and limited VMM parallelism, the VMM output current can be expressed as

$$I_j = \sum_{k}^{m/l} f_{\text{out}}\left(Q_{\text{out}}\left(\sum_{i}^{l} V_i\left(G_{ij}^p - G_{ij}^n\right)\right)\right), \tag{15}$$

where l is the parallelism per VMM computation and m/l is the number of partial sums.

The final VMM results Y_j of forward propagation can be obtained by dividing proportional coefficients K_G and K_V.

$$Y_j = \frac{I_j}{K_G \cdot K_V}. \tag{16}$$

During the backpropagation of HWA training, the above nvCIM nonidealities have not been considered when implementing gradient descent and weight update. For the problem that hardware aware training cannot converge due to larger conductance variation and higher array defect proportion, the method of progressive variation/defect addition can be used to solve the problem.

3.3 Weights mapping correction

Since the evaluation of IR drop induced by interconnect resistance and driving/sensing parasitic resistance is time-consuming, it is costly to inject IR drop into the hardware aware training process [31, 34]. Therefore, we propose WMC [33] to correct weights in advance, so that equivalent weights are closer to the target weights even considering interconnect resistance.

We adopt the WMC method to compensate for weight deviations ahead of the weights programming phase. Firstly, the equivalent weights (G^e) are evaluated after mapping the target conductance (G) to the NVM array with driving/sensing/wire resistance [39].

$$G_{ij}^e = h(G_{ij}, R_{\text{wire}}, R_d, R_s). \tag{17}$$

Error matrix ($K_{\text{IR}} = G^e/G$) is introduced to indicate the conductance deviations. $\mu(K_{\text{IR}})$ is the mean of the error matrix. $\sigma(K_{\text{IR}})$ is the standard deviation of the error matrix. The corrected weights (G^c) are updated as follows:

$$G^c = G + lr \cdot \left(\mu\left(G^e/G\right) \cdot G - G^e\right). \tag{18}$$

By iterating the above correcting process, the standard deviation of the error matrix $\sigma(K_{\text{IR}})$ is tightened to realize weight correction. We program corrected weights (G^c) rather than target conductance (G) to the NVM array. So, Eq. (11) is rewritten as

$$G_{ij} = g\left(f_w\left(Q_{\text{nvm}}\left(G_{ij}^c\right), \sigma_w\right), G_{\text{saf0}}, G_{\text{saf1}}\right). \tag{19}$$

Thus, the equivalent weights (G^e) and the target conductance (G) are approximately proportional to the coefficient $\mu(K_{\text{IR}})$, which can be expressed as

$$G_{ij}^e = h\left(G_{ij}^c, R_{\text{wire}}, R_d, R_s\right) \approx \mu(K_{\text{IR}}) G_{ij}. \tag{20}$$

Therefore, the target output can be obtained after a proportional conversion of the actual output.

$$Y_j = \frac{I_j}{K_G \cdot K_V \cdot \mu(K_{\text{IR}})}. \tag{21}$$

3.4 Mitigating method algorithm

A mitigating method algorithm is composed of HWA module and WMC module, as shown in Algorithm 1. The mitigation algorithm fully considers the impact of analog devices, arrays and mixed-signal circuits on inference accuracy. The mitigation algorithm can also recover the accuracy degradation induced by the above nonidealities effectively and efficiently. We integrate the mitigation algorithm into PyTorch wrapper. In the HWA module, we insert nvCIM nonidealities with low complexity into VMM computation according to (11), (14), and (15). We call PyTorch library functions torch.nn.conv2d/linear to perform convolution and fully-connected operations efficiently. In the WMC module, we evaluate the impact of IR drop according to (17) and compensate for the conductance deviation according to (18). During the training process, only the HWA module is called to re-train DNN models to actively adapt quantization, variation, array defect and nonlinearity of nvCIM accelerators. During the test process, the WMC module is called to correct weight deviation induced by IR drop.

4 Results of mitigating method

To illustrate the effectiveness of the proposed mitigating method, we simulated the mitigating effects of different non-ideal characteristics using the HWA method and the WMC method. Furthermore, to validate the broad generalization of the proposed mitigating method, we conducted simulation tests on larger ImageNet datasets and transformer models.

4.1 Results of various nvCIM non-idealities

The DNN model and dataset used in the simulation are VGG11 and CIFAR10 respectively. The algorithm-level accuracy of the pre-trained VGG11 model for the CIFAR10 is 92.1%.

Quantization. To explore the impact of quantization precision on inference accuracy, we set one of the input and output quantization precision to be 3–8 bits while setting the precision of the other factor to be 8 bits. Figures 5(a) and (b) show that more than 2% inference accuracy is degraded without the HWA method even if the input and weight both are 8 bits. After adopting the HWA method, almost no accuracy loss for the same quantization precision. The simulation results also show that the inference accuracy is more sensitive to input (activation) quantization precision than to weight quantization precision. Activations, influenced by varying input data and network states, tend to exhibit a broader

Algorithm 1 Mitigating method algorithm for DNNs.

Require: DNN models(weight, input);
Require: Non-idealities ($Q_{\rm out}$, $Q_{\rm in}$, $Q_{\rm nvm}$, $f_{\rm in}$, $f_{\rm out}$, $f_{\rm NL}$, $f(\sigma)$, $g(G_{\rm SAF})$, $R_{\rm wire}$, R_d, R_s);
if train == True **then**
  output = **HWA**(weight, input);
  loss = criterion(output, target);
  weight.update(loss);
else if test == True **then**
  weight = **WMC**(weight);
  output = **HWA**(weight, input);
end if
function HWA(weight, input)
  weight = $g(f(Q_{\rm nvm}(\text{weight}), \sigma_w), G_{\rm SAF})$;
  input = $f_{\rm NL}(f_{\rm in}(Q_{\rm in}(\text{input})))$;
  output = conv2d/linear(input, weight);
  output = sum($f_{\rm out}(Q_{\rm out}(\text{output}))$);
  return output;
end function
function WMC(weight)
  cor_weight = $F_{\rm wmc}$(weight, $R_{\rm wire}$, R_d, R_s);
  eqv_weight = h(cor_weight, $R_{\rm wire}$, R_d, R_s);
  return eqv_weight;
end function

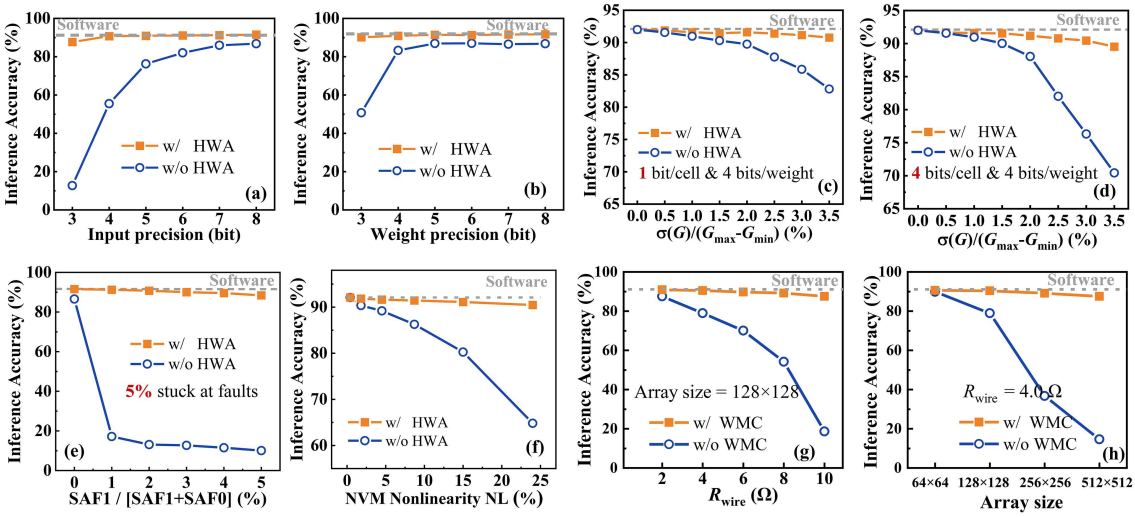

Figure 5 (Color online) Inference accuracy with (a) input quantization, (b) weight precision, (c) device variation for binary conductance, (d) device variation for 4-bit multilevel conductance, (e) different proportions of stuck-at-faults, (f) various *I*-*V* nonlinearity, (g) various wire resistance, and (h) various array sizes.

dynamic range than weights. This necessitates higher quantization precision to accurately represent this range without significant information loss.

Variation. The NVM device variation is assumed to follow Gaussian distribution and device variation is defined as the ratio of the standard deviation of conductance variation to conductance window $\sigma(G)/(G_{\rm max} - G_{\rm min})$. For the 0%–3.5% variation, we simulate the inference accuracy under the condition of binary conductance states per device and 4-bit conductance states per device respectively in Figures 5(c) and (d). Results show that the inference accuracy is more sensitive to multilevel NVM devices than binary NVM devices. For 3.5% variation, the inference accuracy of binary devices and 4-bit devices without the HWA method is reduced by 10% and 22%. After adopting the HWA method, the inference accuracy is recovered efficiently. During the training process, measuring the experiment result of the shallower layer as the input of the deeper layer to retrain the convolutional neural network (CNN) model can effectively mitigate the impact of input-independent device variation on CNN accuracy.

Defect. Devices in SAF0 or SAF1 state can only represent the minimum or maximum weight. Figure 5(e) shows the impact of different SAF0 and SAF1 defect proportions on inference accuracy under the condition of total 5% stuck at faults. Results show that SAF1 has a greater impact on inference accuracy than SAF0. Even only 0.05% SAF1 defects would cause a destructive inference accuracy degradation of nvCIM-based DNNs. Fortunately, by employing the HWA method, the inference accuracy of nvCIM-

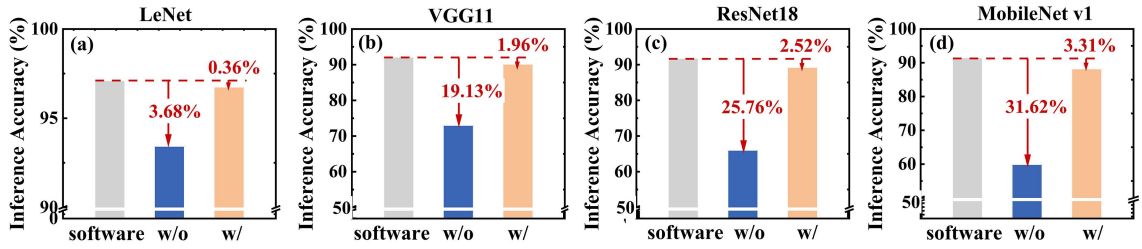

Figure 6 (Color online) Inference accuracy of nvCIM-based DNNs with and without mitigating method. (a) LeNet model for MNIST dataset; (b) VGG11 model for CIRAR10 dataset; (c) ResNet 18 model for CIRAR10 dataset; (d) MobileNet v1 model for CIRAR10 dataset.

based DNNs can be recovered as same as software inference accuracy even for the NVM array with a high proportion of SAF1.

Nonlinearity. The analog input scheme shows great potential in computation delay and energy consumption compared with the digital input scheme [21,35]. The flash linear region is widely utilized in NOR-flash based computing-in-memory chips for the following two considerations: Firstly, the current in the linear region is relatively low, thus the energy consumption of both the flash array and the peripheral readout circuits is reduced. Secondly, when operating flash in the linear region, VMM between multi-bit input and weight can be implemented within a single operational cycle by converting multi-bit input into source-drain voltage using DAC circuits. The I_d-$V_{\rm ds}$ response of flash is nonlinear, and the nonlinearity increases with higher drain voltages. Before adopting the HWA method, we measure the I_d-$V_{\rm ds}$ curve of multilevel flash and fit I_d-$V_{\rm ds}$ curve by a quadratic polynomial. We insert 0.4%–24% nonlinearity into the HWA training process. The inference accuracy of nvCIM-based DNNs with and without HWA training is shown in Figure 5(f). Even if nonlinearity is equal to 24%, the inference accuracy drop of nvCIM-based DNNs with the HWA method is only 1.63%.

IR drop. To show the effectiveness of the WMC method in compensating IR drop, we simulate the inference accuracy of nvCIM-based DNNs with and without the WMC method for 1–10 Ω wire resistance and 4–256 kB array size, respectively, as shown in Figures 5(g) and (h). We set the minimum resistance and maximum resistance of the multilevel NVM device as 10^5 and 10^7 Ω. When exploring the impact of wire resistance, the array size is set to 128 × 128. Figure 5(g) shows that the WMC method can significantly mitigate the inference accuracy degradation caused by wire resistance. In addition, we study the sensitivity of inference accuracy on NVM array size when the wire resistance is 4.0 Ω. With the increase in array size, the inference accuracy degrades more obviously. Figure 5(h) shows that the WMC method can rescue 95% algorithm inference accuracy even if the array size is 512 × 512.

4.2 Results of various DNN models

To prove the effectiveness of the mitigating method on various DNN models, we apply the method to four typical DNN models, namely LeNet-5, VGG11, ResNet18 and MobileNet v1. In these models, the network layers and the weight parameter amounts are significantly different. The simulation setup of these DNN models is 8-bit input and 8-bit weight quantization precision, 2% $\sigma(G)/(G_{\rm max}-G_{\rm min})$ variation, 1% SAF0 and 0.02% SAF1, 2 Ω interconnect resistance $R_{\rm wire}$ (NVM array with 10^5 Ω$R_{\rm on}$ and 10^7 Ω$R_{\rm off}$), and 8.7% NVM nonlinearity. Figure 6 shows the inference accuracy of the LeNet model is reduced slightly due to the high robustness of the LeNet to simple MNIST recognition tasks. For the more complex recognition task CIFAR10, the inference accuracy of MobileNet v1 degrades more seriously than that of ResNet18 and VGG11. This is because computation errors accumulate layer by layer during the chain forward propagation. Directly transferring DNN models trained on GPUs to nvCIM chips for inference is infeasible, as all hardware non-ideal factors would significantly reduce the accuracy of DNN models. When deploying DNN models on nvCIM chips, hardware aware training and WMC are essential. With the mitigating method, there is only 0.36%, 1.96%, 2.52%, and 3.31% accuracy degradation compared with algorithm inference accuracy. Compared to quantization, variation, and defects, the non-linear response of devices and array IR drop has a significant impact on accuracy, as they are related not only to weights but also to inputs. Therefore, the mitigating method can recover 95% software inference accuracy.

To fully validate the effectiveness of the proposed mitigation method, Figures 7(a) and (b) illustrate the accuracy of larger ImageNet datasets and attention-based Transformer models to explore the recovery

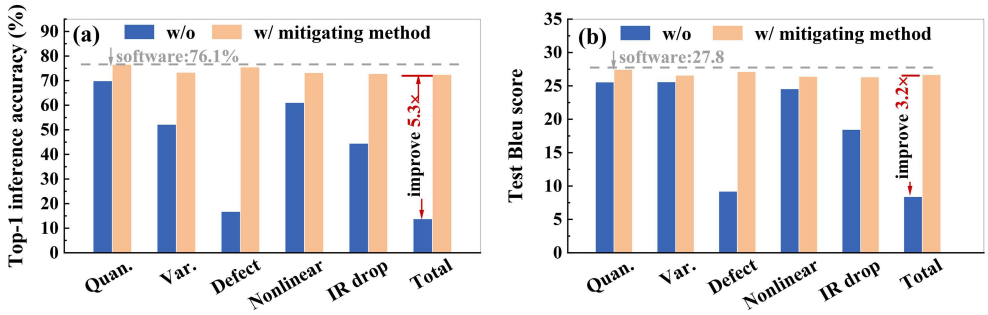

Figure 7 (Color online) Inference accuracy of nvCIM-based DNNs with and without mitigating method. (a) ResNet50 model for ImageNet recognition tasks; (b) transformer model for WMT 2018 Chinese-English translation.

effects of the mitigating method. Figure 7(a) shows the results of the mitigating method on the ImageNet recognition [40] task using the ResNet50 model [41]. Transferring the CNN model trained by GPU to nvCIM chips to conduct inference, all nonidealities cause serious top-1 accuracy drop. Taking all nonidealities into account, the proposed method can recover 95% top-1 accuracy. Figure 7(b) shows the results of WMT 2018 Chinese-English translation[1)] using the Transformer model [42]. Feed-forward layers in the transformer model are implemented in the nvCIM chip. Generating query, key, and value computation in multi-head attention layers is also implemented in the nvCIM chip. Taking all nonidealities into account, the proposed can achieve 3.2 times Test Bleu score improvement.

5 Hardware demonstrations based on NOR flash

Based on the principle and simulation of the mitigating method, we build a five-layer convolutional neural network based on the NOR flash memory computing chip [10] to experimentally illustrate the effectiveness of the proposed mitigation method.

5.1 Experiment setup

Flash-based nvCIM chip. To validate the proposed mitigating method, we fabricated a nvCIM chip with 65 nm NOR flash technology. The test board for the packaged chip and the design under test nvCIM chip layout are shown in Figures 8(a) and (b). Figure 8(c) shows the NOR flash array structure, the weights are programmed into the threshold voltage and the input is applied into word lines (WL). The threshold voltage of the flash transistor is modulated by electron numbers in the floating gate, which are injected from the channel or swept out by tunneling. Figure 8(d) shows the 4-bit conductance distribution using the dynamic drain-voltage programming method [10]. For each state, we randomly select 50 devices for test. The current range is from 0 to 30 μA for 4-bit weight representation.

CNN model and datasets. We compress the LeNet-5 model and clip the input MNIST images, as shown in Figure 9. The Modified CNN model consists of two convolution layers, two pooling layers and one fully connected layer. We quantify float point weights to 4-bit weights uniformly. The original images (28×28) of the MNIST dataset are center-clipped to obtain the 22×22 input images. Each pixel of the cropped image is then quantified from 8 bits (0–255) to 1 bit (0/1). In addition, the feature maps of the middle layers are also quantified to 4 bits uniformly.

Weight mapping and dataflow. The quantized weights of all CNN layers are programmed into two NVM arrays to represent positive and negative weights respectively. The quantized input window of the MNIST image is converted into the input voltage and applied to weight arrays from the most significant bit to the least significant bit. The output current of the positive array and negative array is subtracted and then quantified to obtain the final result. The input images are transferred to the NOR flash memory computing chip controlled by the MCU. The VMM computation of Conv.1, Conv.2 and FC is implemented in NOR flash memory computing chip. The subtraction between output positive and negative current, ReLU activation function and average pooling function are implemented in the CPU.

1) https://statmt.org/wmt18/translation-task.html.

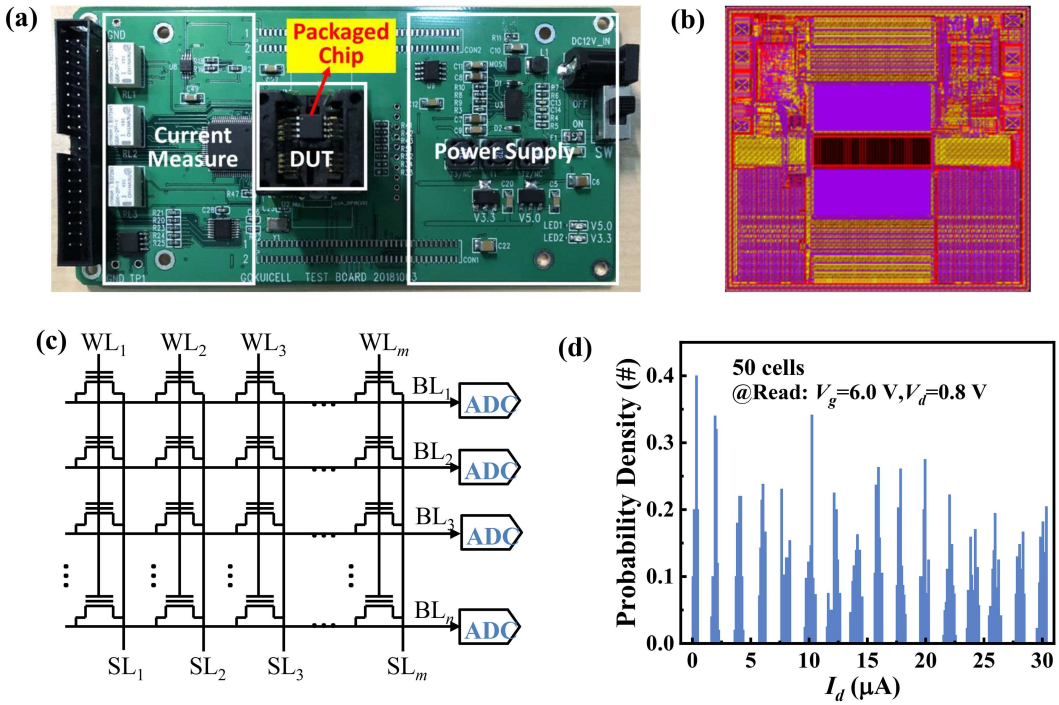

Figure 8 (Color online) (a) The test board for the packaged chip; (b) the design under test nvCIM chip layout; (c) the NOR flash array structure; (d) the multi-bit programming characteristic.

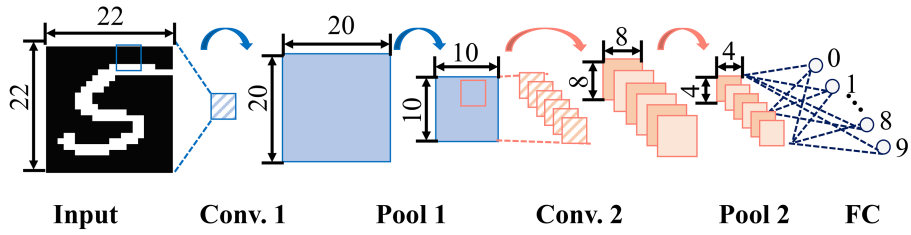

Figure 9 (Color online) Modified LeNet model in NOR flash-based DNN recognition system.

5.2 Experiments of mitigating method

We measure the NOR flash computing chip and extract corresponding characteristic parameters, as shown in Table 3. For the first layer, the input is quantized into binary and the weight is quantized into 4 bits. For the other layers, both the input and weight are quantized into 4 bits. We adopt the dynamic drain-voltage programming method to program multi-bit flash. Then we measure the drain current distribution of flash devices with 4-bit conductance states. The current variation is less than 1.5%. Since the proportion of SAF devices in the NOR flash memory computing chip is approximately zero, to illustrate the effectiveness of the mitigation method on array defect, we assume that the proportion of SAF0 and SAF1 defects in the flash array is 5% and 0.5% respectively, and they are randomly distributed in the flash array, as shown in Figure 10. By comparing the measured output current with the ideal output current for various NVM conductance, we derive that the equivalent sensing resistance of the NOR flash computing chip is 2934 Ω.

In the following hardware aware training, the input/weight/output quantization, flash variation and array defect are injected into the pre-trained modified LeNet model. The loss function is reduced through 30 training epochs to enable the re-trained DNN model to actively adapt nonidealities of the NOR flash memory computing chip. Then, the weights are iteratively corrected by the WMC method considering the sensing resistance. Finally, the corrected weights are programmed into the NOR flash memory computing chip to recognize the MNIST dataset.

Table 3 Characteristic parameters of NOR-flash memory chip.

	nvCIM hardware parameters	Value
Quantization	$Q_{\text{in}}(V), \omega_{\text{in}}$ $Q_{\text{nvm}}(G), \omega_w$ $Q_{\text{out}}(I), \omega_{\text{out}}$	V = 0.8 V, ω_{in} =1 bit (L1) ω_{in} =4 bits (L2–L5) G = 0–37.5 μS, ω_w = 4 bits ω_{out} = 4 bits
Variation	$f_w(G, \sigma_w)$ ω_w, σ, N	Gaussian distribution ω_w = 4 bits, σ = 1.5%, N = 4 bits
Array defect	$g(G, G_{\text{saf0}}, G_{\text{saf1}})$	Figure 10
IR drop	R_s	2934 Ω

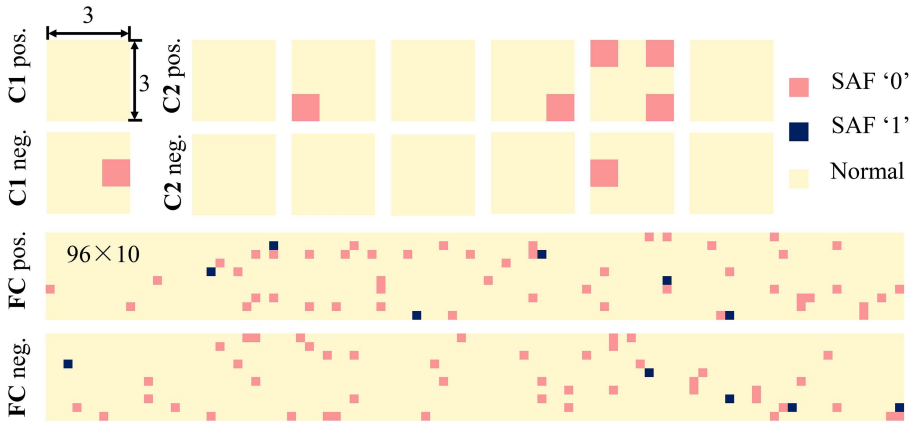

Figure 10 (Color online) Defect distribution (5% SAF0, 0.5% SAF1) in NOR flash array.

5.3 Experiment results

Figure 11 shows the recognition process of an image (number '5') with the NOR flash based neural network recognition system. The image is randomly selected from the cropped and quantified MNIST dataset. Convolutional layers Conv.1 and Conv.2, including 1 and 6 convolutional kernels respectively. In the process of digit recognition, binary input is fed into an NOR-flash chip to implement the Conv.1 layer. The output current corresponding to positive and negative weights is measured. After quantization and the pooling layer, the 4-bit feature maps (Bit3–Bit0) are input into the Conv.2 layer in a bit-wise manner. Subsequently, the output current of the Conv.2 layer corresponding to positive and negative weights is again measured. Following quantization and another pooling, the 4-bit feature maps (Bit3–Bit0) are fed into the FC layer. The output of the FC layer is not further quantized; instead, the output currents are directly compared. The node corresponding to the maximum value is identified as the recognition result. As shown in Figure 11, the maximum value of FC_Out is 5, which indicates that the input image is classified as the number '5'.

We measure and count the inference accuracy of 10000 MNIST test images, as shown in Figure 12. If the original weights are programmed to NOR flash memory computing chip without the mitigating method, the measured inference accuracy is only 82.12%. There is 14.98% accuracy degradation compared with algorithm inference accuracy (97.1%). Then, the re-trained and corrected weights are programmed into the NOR flash computing chip. With the proposed mitigating method, the measured inference accuracy is 96.69%, which is only 0.41% lower than the algorithm inference accuracy.

The experimental demonstration results also prove that the proposed mitigating method including hardware aware training and weights mapping correction can effectively recover the inference accuracy degradation induced by nonidealities of nvCIM accelerator across circuit, array and device.

6 Conclusion

In this paper, a mitigating method is developed to narrow the accuracy gap between the nvCIM based DNNs and the software DNNs. We establish the hardware characteristic behavior model to analyze the impact of nvCIM nonidealities on inference accuracy. According to the computational complexity of nvCIM characteristics, we propose hardware aware training and weights mapping correction to re-train

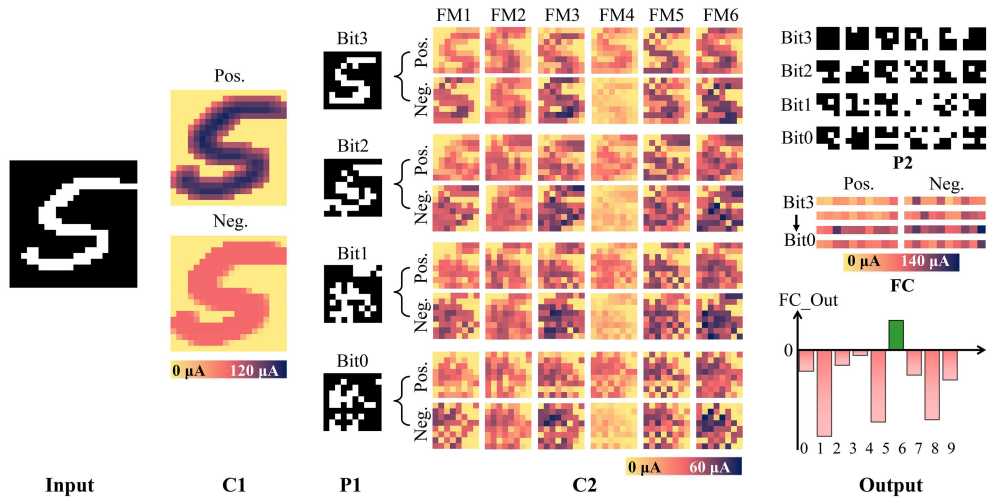

Figure 11 (Color online) Experimental demonstration of the recognition of number '5' by convolution neural network based on flash memory computing chip with mitigating method.

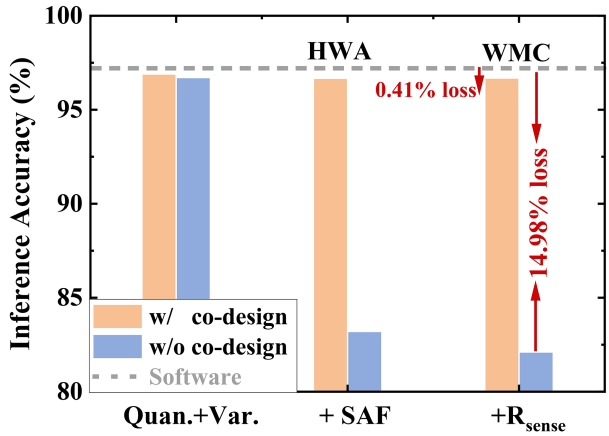

Figure 12 (Color online) Measured inference accuracy of NOR flash based neural network with and without mitigating method.

and correct DNN weights to mitigate the impact of nvCIM hardware nonidealities on the inference accuracy respectively. Simulation and demonstration results show that the proposed mitigating method can effectively alleviate the inference accuracy degradation without increasing hardware overheads. Moreover, the proposed mitigating method can be migrated to other VMM-based algorithms implemented by crossbar-based hardware accelerators.

Acknowledgements This work was supported in part by National Key R&D Program of China (Grant No. 2023YFB4402405), National Natural Science Foundation of China (Grant Nos. 92064001, 62101018), 111 Project (Grant No. B18001), and Joint Funds of the National Natural Science Foundation of China (Grant No. U20A20204).

References

1 Yao P, Wu H, Gao B, et al. Fully hardware-implemented memristor convolutional neural network. Nature, 2020, 577: 641–646
2 Wan W, Kubendran R, Schaefer C, et al. A compute-in-memory chip based on resistive random-access memory. Nature, 2022, 608: 504–512
3 Xue C X, Hung J M, Kao H Y, et al. A 22nm 4Mb 8b-precision ReRAM computing-in-memory macro with 11.91 to 195.7 TOPS/W for tiny AI edge devices. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), 2021. 245–247
4 Hung J M, Huang T H, Huang S P, et al. An 8-Mb DC-current-free binary-to-8b precision ReRAM nonvolatile computing-in-memory macro using time-space-readout with 1286.4-21.6 TOPS/W for edge-AI devices. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), 2022
5 Song S Y, Huang P, Shen W S, et al. A 3.3-Mbit/s true random number generator based on resistive random access memory. Sci China Inf Sci, 2023, 66: 219402
6 Khaddam-AljamehR, Stanisavljevic M, Fornt Mas J, et al. HERMES core-A 14nm CMOS and PCM-based in-memory compute core using an array of 300ps/LSB linearized CCO-based ADCs and local digital processing. In: Proceedings of Symposium on VLSI Circuits, 2021
7 Khwa W S, Chiu Y C, Jhang C J, et al. A 40-nm, 2M-Cell, 8b-precision, hybrid SLC-MLC PCM computing-in-memory Macro with 20.5-65.0 TOPS/W for tiny-Al edge devices. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), 2022

8 Chiu Y C, Yang C S, Teng S H, et al. A 22nm 4Mb STT-MRAM data-encrypted near-memory computation Macro with a 192GB/s read-and-decryption bandwidth and 25.1-55.1 TOPS/W 8b MAC for AI operations. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC), 2022
9 Guo Z, Yin J, Bai Y, et al. Spintronics for energy-efficient computing: an overview and outlook. Proc IEEE, 2021, 109: 1398–1417
10 Xiang Y C, Huang P, Yang H Z, et al. Storage reliability of multi-bit flash oriented to deep neural network. In: Proceedings of IEEE International Electron Devices Meeting (IEDM), 2019
11 Zhang D, Wang H, Feng Y, et al. Implementation of image compression by using high-precision in-memory computing scheme based on NOR flash memory. IEEE Electron Dev Lett, 2021, 42: 1603–1606
12 Yu G H, Huang P, Han R Z, et al. Co-optimization strategy between array operation and weight mapping for flash computing arrays to achieve high computing efficiency and accuracy. Sci China Inf Sci, 2023, 66: 129403
13 Yang H Z, Huang P, Han R Z, et al. An ultra-high-density and energy-efficient content addressable memory design based on 3D-NAND flash. Sci China Inf Sci, 2023, 66: 142402
14 Yayla M, Thomann S, Buschjager S, et al. Reliable binarized neural networks on unreliable beyond von-Neumann architecture. IEEE Trans Circ Syst I, 2022, 69: 2516–2528
15 Soliman T, Müller F, Kirchner T, et al. Ultra-low power flexible precision FeFET based analog in-memory computing. In: Proceedings of IEEE International Electron Devices Meeting (IEDM), 2020
16 Jeong Y J, Zidan M A, Lu W D. Parasitic effect analysis in memristor-array-based neuromorphic systems. IEEE Trans Nanotechnol, 2018, 17: 184–193
17 Chen L R, Li J W, Chen Y R, et al. Accelerator-friendly neural-network training: learning variations and defects in RRAM crossbar. In: Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017
18 Woo J, Moon K, Song J, et al. Improved synaptic behavior under identical pulses using AlO_x/HfO_2 bilayer RRAM array for neuromorphic systems. IEEE Electron Dev Lett, 2016, 37: 994–997
19 Mao R, Wen B, Jiang M, et al. Experimentally-validated crossbar model for defect-aware training of neural networks. IEEE Trans Circ Syst II, 2022, 69: 2468–2472
20 Li H, Jiang Z, Huang P, et al. Variation-aware, reliability-emphasized design and optimization of RRAM using SPICE model. In: Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015
21 Xiang Y C, Huang P, Zhou Z, et al. Analog deep neural network based on nor flash computing array for high speed/energy efficiency computation. In: Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), 2019
22 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, 2018
23 Sun X, Yu S. Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional neural networks. IEEE J Emerg Sel Top Circ Syst, 2019, 9: 570–579
24 Jain S, Sengupta A, Roy K, et al. RxNN: a framework for evaluating deep neural networks on resistive crossbars. IEEE Trans Comput-Aided Des Integr Circ Syst, 2021, 40: 326–338
25 Peng X, Huang S, Jiang H, et al. DNN+NeuroSim V2.0: an end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training. IEEE Trans Comput-Aided Des Integr Circ Syst, 2021, 40: 2306–2319
26 Zhu Z, Sun H, Xie T, et al. MNSIM 2.0: a behavior-level modeling tool for processing-in-memory architectures. IEEE Trans Comput-Aided Des Integr Circ Syst, 2023, 42: 4112–4125
27 Chakraborty I, Ali M F, Kim D E, et al. GENIEx: a generalized approach to emulating non-ideality in memristive Xbars using neural networks. In: Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC), 2020
28 Liu C C, Hu M, Strachan J P, et al. Rescuing memristor-based neuromorphic design with high defects. In: Proceedings of the 54th Annual Design Automation Conference, 2017
29 Yan B N, Yang J H, Wu Q, et al. A closed-loop design to enhance weight stability of memristor based neural network chips. In: Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2017
30 Liu B Y, Hu M, Li H, et al. Digital-assisted noise-eliminating training for memristor crossbar-based analog neuromorphic computing engine. In: Proceedings of the 50th ACM/EDAC/IEEE Design Automation Conference (DAC), 2013
31 He Z Z, Lin J, Ewetz R, et al. Noise injection adaption: end-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping. In: Proceedings of the 56th Annual Design Automation Conference, 2019
32 Chen P Y, Lin B B, Wang I T, et al. Mitigating effects of non-ideal synaptic device characteristics for on-chip learning. In: Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2015
33 Han L X, Xiang Y C, Huang P, et al. Novel weight mapping method for reliable NVM based neural network. In: Proceedings of IEEE International Reliability Physics Symposium (IRPS), 2021
34 Liao Y, Gao B, Yao P, et al. Diagonal matrix regression layer: training neural networks on resistive crossbars with interconnect resistance effect. IEEE Trans Comput-Aided Des Integr Circ Syst, 2021, 40: 1662–1671
35 Sung C, Lim S, Kim H, et al. Effect of conductance linearity and multi-level cell characteristics of TaO_x-based synapse device on pattern recognition accuracy of neuromorphic system. Nanotechnology, 2018, 29: 115203
36 Liu P, You Z, Wu J, et al. Fault modeling and efficient testing of memristor-based memory. IEEE Trans Circ Syst I, 2021, 68: 4444–4455
37 Chen C Y, Shih H C, Wu C W, et al. RRAM defect modeling and failure analysis based on march test and a novel squeeze-search scheme. IEEE Trans Comput, 2015, 64: 180–190
38 Chen A. A comprehensive crossbar array model with solutions for line resistance and nonlinear device characteristics. IEEE Trans Electron Dev, 2013, 60: 1318–1326
39 Han R Z, Huang P, Zhao Y D, et al. Efficient evaluation model including interconnect resistance effect for large scale RRAM crossbar array matrix computing. Sci China Inf Sci, 2019, 62: 22401
40 Deng J, Dong W, Socher R, et al. ImageNet: a large-scale hierarchical image database. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2009
41 He K M, Zhang X Y, Ren S Q, et al. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016
42 Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017