How Do Errors Impact NN Accuracy on Non-Ideal Analog PIM? Fast Evaluation via an Error-Injected Robustness Metric

Lidong Guo[1*], Zhenhua Zhu[1*†], Qiushi Lin[1], Yuan Xie[2], Huazhong Yang[1], Wangyang Fu[1†], Yu Wang[1†]
[1]Tsinghua University, Beijing, China [2]HKUST, Hong Kong, China
[*]Both authors contributed equally to this research.
[†]Corresponding authors: {zhuzhenhua, fwy2018, yu-wang}@tsinghua.edu.cn

***Abstract*—The emerging analog Processing-in-Memory (PIM) architectures have shown great potential to overcome the memory wall problem and accelerate neural network (NN) inference. However, different from digital architectures, the computation accuracy of analog PIM architectures is directly impacted by various errors, which are related to both software and hardware parameters. Existing PIM simulators mainly adopt the bit-and-crossbar slicing paradigm to evaluate the accuracy under various errors. Each MVM operation is performed bit by bit and crossbar by crossbar, which is extremely time-consuming, especially for models with a larger number of parameters, such as large language models (LLMs).**

In this work, we propose an error-injected robustness metric, unifying various errors into the weight dimension and facilitating joint error analysis. Based on the error-injected robustness metric, we propose a Non-Ideal PIM Accuracy (NIPA) evaluation model for relative accuracy evaluation, considering the coupling effect (i.e., various errors can be affected by the same factor) among various errors using NN's prior information. We further propose a non-slicing absolute accuracy evaluation method, eliminating the need for the time-consuming bit-and-crossbar slicing process. Extensive experiments on CNNs and LLMs validate that the proposed NIPA evaluation model achieves high correlations of up to 0.91 with the absolute accuracy evaluated by DNN+NeuroSim. At the same time, compared to existing bit-and-crossbar slicing evaluation methods, the proposed non-slicing absolute accuracy evaluation method achieves up to 105.8× speedup with average evaluation errors as low as 0.29%.

***Index Terms*—non-volatile memory device, non-ideal factor, processing-in-memory, accuracy evaluation**

I. Introduction

Neural networks (NN) have shown remarkable performance in various fields [1]–[3]. However, as the number of NN parameters continues to grow, traditional von Neumann architectures (e.g., CPU and GPU) encounter the memory wall problem. During NN inference, massive data moves between separate memory storage and computing processors, resulting in high power consumption and long latency.

The emerging analog Processing-In-Memory (PIM) architectures based on non-volatile memory (NVM) devices have shown the potential to eliminate data movements and improve hardware performance. However, due to its in-situ matrix-vector multiplication (MVM) in the analog domain, the computation accuracy is directly impacted by various errors, including weight quantization error, analog-to-digital conversion error, and device error, which are tightly coupled and closely related to both software and hardware parameters.

To evaluate the impact of various errors on the accuracy of analog PIM architectures, existing PIM simulators including DNN+NeuroSim [4], MNSIM 2.0 [5], DL-RSIM [6], Swordfish [7], Geniex [8], etc., [9]–[16] mainly adopt the bit-and-crossbar slicing paradigm to inject various errors during NN inference process. Specifically, depending on the chosen weight bit-width and crossbar size, a weight matrix needs to be split into multiple matrices along the bit dimension and sliced into smaller matrices to fit the crossbar size. Then, each MVM operation in the neural network is then performed bit by bit and crossbar by crossbar, with device conductance variation and ADC conversion error injected. Based on the above simulation paradigm, the absolute accuracy of the neural network on analog PIM architectures can be accurately evaluated, facilitating the analysis of various errors' impacts on accuracy [17] and further optimization of algorithmic performance on the analog PIM architectures [9], [18]–[22].

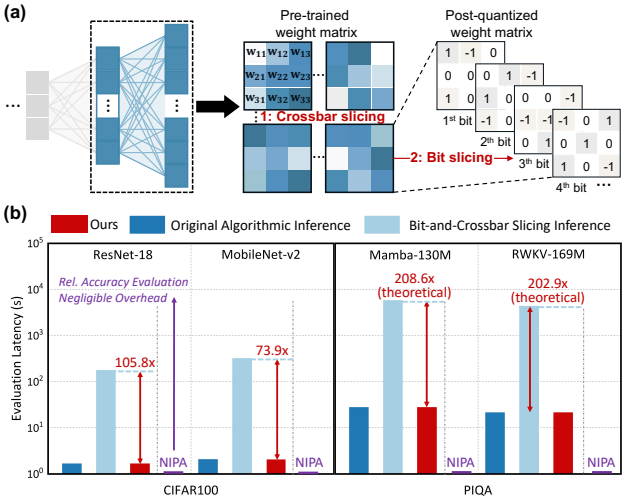

Fig. 1: Simulating NN inference through bit-and-crossbar slicing incurs intolerable evaluation overhead.

Although such an accuracy evaluation scheme is consistent with how the analog PIM architectures actually perform the calculation, it requires more MVM operations than the algorithmic inference process, incurring additional evaluation overhead. As shown in Fig. 1, simulated through the bit-and-crossbar-slicing scheme, the simulation latency increases more than 100 times. As the number of model parameters or dataset size grows, the accuracy evaluation overhead becomes intolerable (e.g., >10,000 seconds for an end-to-end accuracy evaluation process). Furthermore, compared to the separate analysis for each type of error, it is more challenging to

TABLE I: Comparison of typical non-ideal PIM accuracy evaluation methods.

	Abs. Acc	Rel. Acc	Coupling Effect	Error-injected Dim.	Models	Non-ideal Factors
DNN+NeuroSim [4]	Slow	✖	Strong	Device/Psum-level	CNN	Quant./ADC/Device*
MNSIM2.0 [5]	Slow	✖	Strong	Device/Psum-level	CNN	Quant./ADC/Device*
DL-RSIM [6]	Slow	✖	Weak	Psum-level	CNN	Quant./Device*
Geniex [8]	Slow	✖	Strong	Device/Psum-level	CNN	Quant./ADC/Xbar
PytorX [9]	Slow	✖	Weak	Device-level	CNN	Crossbar/Device*
MICSim [10]	Slow	✖	Weak	Device/Psum-level	CNN/LLM	Quant./ADC
CoMN [11]	Slow	✖	Strong	Device/Psum-level	CNN	Quant./ADC/Xbar/Device*
Swordfish [7]	Slow	✖	Strong	Device/Psum-level	Basecaller	Quant./ADC/Device*
RxNN [23]	Slow	✖	Strong	Device/Psum-level	CNN	Quant./ADC/Device*
Yan, et al. [12]	Fast	✖	✖	Device-level	CNN	Device*
Gibbon [13]	✖	Predictor	Weak	Device/Psum-level	CNN	Quant./ADC
Unified-QCN [14]	✖	Metric	Weak	Device/Psum-level	CNN	Quant./ADC/Device*
This Work	**Fast**	**Metric**	**Strong**	**Weight-level**	**CNN/LLM**	**Quant./ADC/Xbar/Device***

Device*: different types of device conductance deviation caused by inaccurate programming, Stuck-at-Fault(SAF), conductance retention, and so on.
Strong: comprehensive consideration of coupling effects is achieved by injecting all errors into the slicing computation or directly analyzing their interactions in derivation.
Weak: insufficient consideration of coupling effects due to the incomplete consideration of non-ideal factors or the indirect modeling by NN-based predictor.

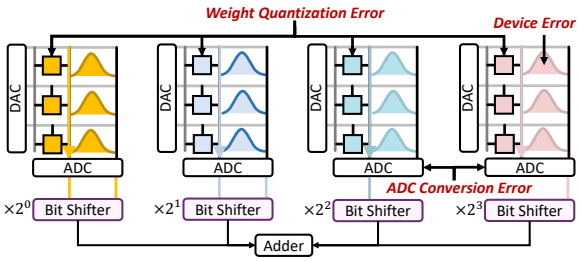

Fig. 2: Analog PIM architecture with errors injected in different dimensions.

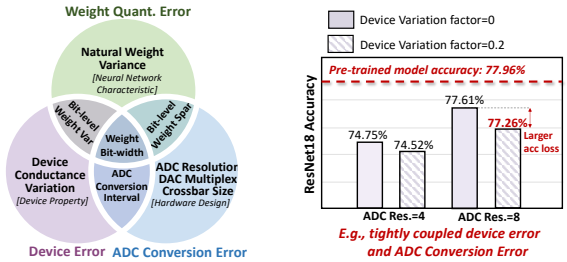

Fig. 3: Complex coupling effect among various errors.

evaluate the combined impacts of different errors due to the intolerable overhead introduced by the enlarged parameter space. To mitigate the accuracy evaluation overhead, existing work evaluates the relative accuracy using either a pre-trained NN-based accuracy predictor [13] or a mathematically derived metric [14]. While these approaches reduce simulation time, they do not support the absolute accuracy evaluation.

In this work, the key objective is to propose an effective and efficient relative & absolute accuracy evaluation method for non-ideal PIM architectures. However, challenges arise when exploring the correlation between various errors and NN inference accuracy on PIM. Firstly, the injected dimensions of various errors are different. As shown in Fig. 2, device error, weight quantization error, and ADC conversion error are injected into single-device, multi-device, and partial sum (psum) dimensions, respectively, making it difficult to jointly analyze their impact on accuracy without bit-and-crossbar slicing. Secondly, complex coupling effects exist among different errors. As Fig. 3 shows, various errors can be affected by the same factor. For example, when the ADC resolution decreases from 8 to 4, the larger conversion interval incurs larger ADC quantization error. At the same time, the impact of accumulated device error is attenuated by the larger conversion interval. In this cases, an incomplete error analysis could impact the effectiveness of the accuracy evaluation.

To address the above challenges, we unify different types of errors into *weight* dimension and propose an error-injected robustness metric. Then, we propose a Non-Ideal PIM Accuracy (NIPA) evaluation model to evaluate the relative NN accuracy on analog PIM architectures, considering the coupling effects using NN's prior information. Additionally, we propose a non-slicing absolute accuracy evaluation method, eliminating the need for time-consuming bit-and-crossbar slicing processes. As shown in Table I, compared to existing PIM accuracy evaluation methods, this work achieves further improvements in evaluation efficiency, comprehensiveness, supported model, and error types. Our key contributions include:

- We propose an error-injected robustness metric, capturing the correlation between original weight variance, post-deployment weight error variance, and NN inference accuracy. By unifying various errors injected in different dimensions into the weight dimension, the proposed robustness metric enables joint error analysis.
- We derive a Non-Ideal PIM Accuracy (NIPA) evaluation model based on the error-injected robustness metric, considering the complex coupling effects using NN's prior information. The NIPA evaluation model enables accurate relative accuracy evaluation and consistently exhibits high correlations of up to 0.91 with the absolute accuracy evaluated by DNN+NeuroSim on both convolutional neural networks (CNNs) and large language models (LLMs).
- We propose a non-slicing absolute accuracy evaluation method. With different errors unified and injected in the weight dimension, the absolute accuracy on the target dataset can be evaluated during the original algorithmic inference process, without the need for bit-and-crossbar slicing. Compared to typical DNN+NeuroSim simulator,

the proposed method achieves up to 105.8× speedup with average evaluation errors as low as 0.29%.

- The proposed evaluation methods facilitate the co-optimization of algorithmic and hardware performance, paving the way for efficient and scalable design of analog PIM architectures. By leveraging the non-slicing paradigm, the proposed methods significantly enhance the exploration efficiency across a broader design space.

II. Preliminary

A. Analog Processing-in Memory Architecture

In analog Processing-in-Memory (PIM) architectures, the weight matrix and input vectors are represented by device conductance $\mathbf{G}$ and voltage vector $\mathbf{V}$ respectively, with MVMs performed using Kirchhoff's and Ohm's laws [24]. Despite the superiority in its in-situ computing mode, they suffer from various errors in the non-ideal analog domain: (1) Existing NVM devices suffer from large conductance variation, due to the limitations of device intrinsic properties and process variation; (2) Weight quantization for neural networks incurs significant accuracy loss; (3) The sliced MVM results need to be digitized by Analog-Digital Converter (ADC). The limited ADC resolution incurs signal precision loss. Orthogonal to existing work that mitigates the impact of various non-ideal factors, this work aims to enhance the efficiency of analog PIM's accuracy evaluation, thus assessing the combined effects of various non-ideal factors efficiently and facilitating further device/algorithm/architecture-level optimization.

B. Accuracy Evaluation for PIM Architecture

Two types of PIM accuracy evaluation methods have been developed to evaluate the accuracy of NNs on non-ideal analog PIM. The first type mainly includes DNN+NeuroSim [4], MNSIM 2.0 [5], DL-RSIM [6], Geniex [8], PytorX [9], Swordfish [7], etc., [7], [10]–[12], [23], [25]–[27], which employ the bit-and-crossbar slicing paradigm for the evaluation of the analog PIM's absolute accuracy across different models. While these methods provide fine-grained absolute accuracy evaluation under various errors, the evaluation overhead increases as the number of NN parameters grows. The second type evaluates relative accuracy through an NN-based predictor [13] or a mathematically derived metric [14]. Although these approaches reduce simulation time, they do not support the absolute accuracy evaluation and do not adequately consider the coupling effects between various errors. In contrast, this work enables efficient evaluation for both absolute and relative accuracy of analog PIM architectures, taking the strong coupling effect among various non-ideal factors into account.

III. Method Overview

Fig. 4 gives an overview of the proposed method. We first propose an error-injected robustness metric to describe the correlation between original weight variance σ_w^2, post-deployment weight error variance σ_{err}^2, and NN relative inference accuracy ACC. By unifying various errors injected in different dimensions into the weight dimension, the proposed

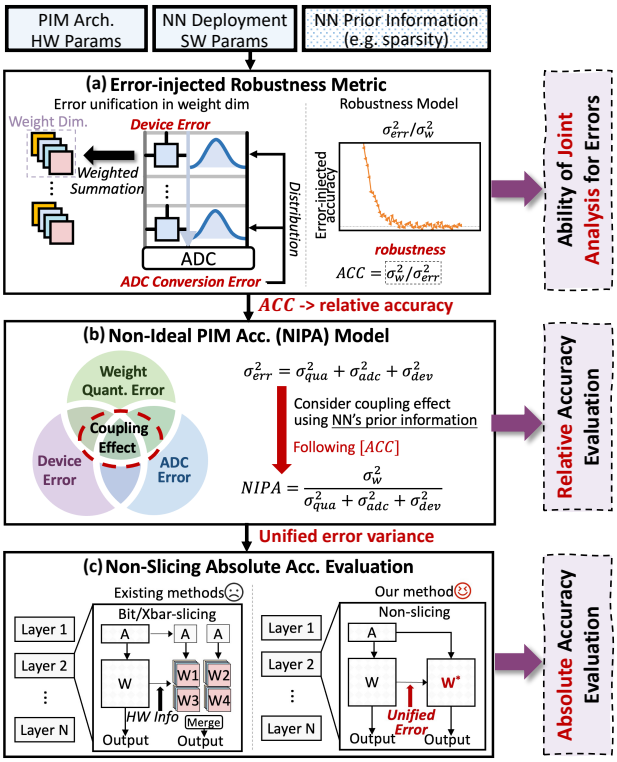

Fig. 4: The proposed non-ideal PIM evaluation method supports both relative/absolute accuracy evaluation, enabling joint analysis for various errors.

robustness metric facilitates joint analysis for various errors. Guided by the error-injected robustness metric, we propose a Non-Ideal PIM Accuracy (NIPA) evaluation model for relative accuracy evaluation, considering the coupling effect among various errors using NN's prior information. We further propose a non-slicing evaluation paradigm based on the insight of unified error variance calculation. As shown in Fig. 4(c), the absolute accuracy can be evaluated through the original algorithmic inference process of the neural network.

IV. Error-Injected Robustness Metric

In NVM-based analog PIM architectures, the computation accuracy is directly impacted by various types of errors. As introduced in Section I, the injected dimensions of various errors are different, making it difficult to jointly analyze their impact on accuracy without bit-and-crossbar slicing.

Here, **our fundamental insight is that errors injected in different dimensions can be mapped to the weight dimension through distribution and weighted summation**, as Fig. 4(a) shows. Firstly, based on the current accumulation characteristics and device equivalence assumption, the ADC conversion error E_{adc} in the partial sum dimension can be evenly distributed to each device along the column direction:

$$E_{adc\_dev} = \frac{E_{adc}}{S}, \tag{1}$$

where E_{adc_dev} are distributed ADC conversion errors on each device and S is the crossbar size. Secondly, before being deployed on analog PIM architecture, NN's weights need to

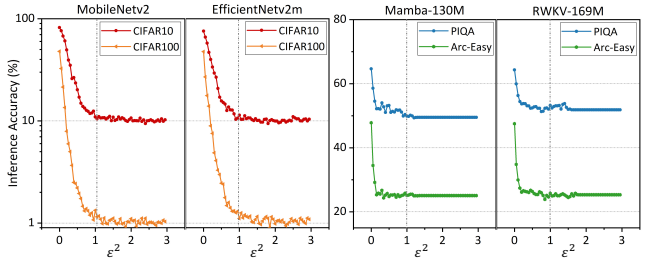

Fig. 5: NN inference accuracy completely collapses once injected error variance reaches original weight variance.

be quantized following $w = s * w_{int}$, where s is the scaling factor. Assuming the single-level NVM devices, a b_w-bit weight value w is represented by b_w NVM devices following $w = s * \sum_{i=0}^{b_w-1} 2^i w_i$, where w_i is the bit-level weight value represented by the device conductance. Therefore, the device error from multiple devices corresponding to the same weight value will accumulate. The accumulated error in the weight dimension E_{dev_weight} is calculated as:

$$E_{dev\_weight} = s * \sum_{i=0}^{b_w-1} 2^i \cdot E_{dev_i}. \tag{2}$$

Notably, the ADC conversion error distributed to each device can also be accumulated similar to (2).

Based on the above insight, we can unify weight quantization error E_{qua}, ADC conversion error E_{adc}, and device error E_{dev} into the weight dimension following $E = E_{qua} + E_{adc\_weight} + E_{dev\_weight}$, enabling the joint analysis of various errors without the need for bit-and-crossbar slicing. We conduct an oracle experiment to explore the relationship between weight error E and model accuracy. Given a pre-trained neural network, we inject errors with specific variance onto each weight value and evaluate the inference accuracy on the target dataset. Based on the zero-mean Gaussian error assumption, the error injection process is defined as:

$$\mathbf{W}^* = \mathbf{W} + \mathbf{W} \cdot \mathcal{N}(0, \varepsilon^2), \tag{3}$$

where $\mathbf{W}$ and $\mathbf{W}^*$ are original and error-injected weights, respectively. The latter term represents the unified error E in the weight dimension. $\mathcal{N}(0, \varepsilon^2)$ is Gaussian distribution with variance equal to ε^2. Given that the weights of the pre-trained neural network follow a Gaussian distribution $\mathcal{N}(0, \sigma_w^2)$, the variance of unified error σ_{err}^2 equals $\varepsilon^2 \sigma_w^2$.

As shown in Fig. 5, when ε^2 reaches 1, that is, once the variance of unified error σ_{err}^2 reaches the original weight variance σ_w^2, the inference accuracy of neural network completely collapses, which is consistently observed across different models and datasets. Here, we define an **error-injected robustness metric**, unifying various errors into the weight dimension. The robustness metric describes the correlation between relative computation accuracy ACC, original weight variance σ_w^2, and the post-deployment weight error variance σ_{err}^2 as follows:

$$ACC = \sigma_w^2 / \sigma_{err}^2. \tag{4}$$

$\sigma_w^2/\sigma_{err}^2$ is abstracted to represent the NN's robustness, i.e., the model's ability to maintain stable and reliable performance facing error interference or parameter changes. Referring to the results of mobilenetv2 in Fig. 5, when ACC drops to 2 (ε^2 reaches 0.5), the absolute accuracy on CIFAR10 and CIFAR100 datasets drops to about 20% and 2%, respectively.

V. NIPA: Non-Ideal PIM Accuracy Evaluation Model

The proposed error-injected robustness metric provides a feasible way to unify various errors and evaluate their joint impact on NN's inference accuracy without bit-and-crossbar slicing. As shown in Fig. 4(b), based on the robustness metric, we propose a Non-Ideal PIM Accuracy (NIPA) evaluation model to evaluate the relative computation accuracy of analog PIM architectures with different configurations.

As illustrated in Section IV, multiple NVM devices form the encoded binary number of one weight. Therefore, the original weight variance can be calculated by the weighted summation of each weight bit's variance after being deployed on PIM:

$$\sigma_w^2 = s^2 \cdot \sum_{i=0}^{b_w-1} 2^{2i} \sigma_{w_i}^2. \tag{5}$$

After obtaining the original weight variance, the next step is to derive the unified error variance in the weight dimension. In the following subsections, we will analyze various errors and mathematically derive the NIPA evaluation model based on the unified error variance.

A. Weight Quantization Error

During the quantization process, the weight values within each quantization interval are rounded to the same value. The quantization interval is set as the quantization range divided by the number of quantization levels. Given that 99.74% of the weights are within the $[-3\sigma_w, 3\sigma_w]$ range theoretically, we set the quantization range to $6\sigma_w$. Therefore, the quantization interval Q_{INR} is calculated as:

$$Q_{INR} = 6\sigma_w / 2^{b_w}, \tag{6}$$

where b_w is the adopted weight bit-width. The original weight variance σ_w^2 is obtained in (5). Since the quantization interval remains constant across different weight value ranges, the overall quantization error variance can be approximated as the error variance within the center quantization interval. The corresponding error variance σ_{qua}^2 of E_{qua} can be derived by:

$$\sigma_{qua}^2 = \int_{-\frac{Q_{INR}}{2}}^{\frac{Q_{INR}}{2}} \frac{1}{Q_{INR}} \cdot x^2 dx = 3\sigma_w^2 / 2^{2b_w} \tag{7}$$

B. Weight Information-aware ADC Conversion Error

Similar to weight quantization, the analog-to-digital conversion also involves rounding the values within an interval to the same value. The difference is that such conversion is performed on the partial sum of each crossbar (corresponding to different bit operations), rather than on individual weights. As a result, ADC conversion error is related to both hardware parameters and NN weight information, including bit-level weight sparsity and weight matrix size.

The ADC conversion interval is related to the crossbar size S, ADC resolution a_r, and bit-level weight sparsity sp along the column direction, and is calculated as below:

$$C_{INR_i} = S \cdot (1 - sp_i)/2^{a_r}. \tag{8}$$

C_{INR_i} and sp_i are conversion interval and sparsity corresponding to i-th weight bit, respectively. The sparsity information is obtained from the bit-level statistics of different layers in the neural network. Following the distribution step introduced in Section IV, the ADC conversion error is distributed to each device, and thus the distributed error variance $\sigma_{adc\_dev_i}$ on the device corresponding to i-th weight bit is derived as:

$$\begin{aligned} S \cdot \sigma^2_{adc\_dev_i} &= \int_{-\frac{C_{INR_i}}{2}}^{\frac{C_{INR_i}}{2}} \frac{1}{C_{INR_i}} \cdot x^2 dx = \frac{S^2 \cdot (1 - sp_i)^2}{12 \cdot 2^{2a_r}} \\ \Rightarrow \sigma^2_{adc\_dev_i} &= \frac{S \cdot (1 - sp_i)^2}{12 \cdot 2^{2a_r}} \end{aligned} \tag{9}$$

After the ADC conversion error in the psum dimension is uniformly distributed to each device, it is aligned to the weight dimension by weighted summation:

$$\sigma^2_{adc\_weight} = s^2 \sum_{i=0}^{b_w - 1} 2^{2i} \sigma^2_{adc\_dev_i}. \tag{10}$$

It is important to note that the weight matrix size not only affects the ADC conversion interval but also influences the overall MVM calculation error. When the weight matrix size exceeds crossbar size S, the matrix needs to be sliced, and the ADCs' outputs of all crossbars need to be merged. In this case, ADC conversion errors will be multiplied. Note that when the number of rows that can be driven simultaneously is limited to N, the effective crossbar size S should be adjusted to N. Assume that the weight matrix is divided into k crossbars along the column direction, (9) needs to be changed to:

$$\sigma^2_{adc\_dev_i} = \frac{S \cdot (1 - sp_i)^2}{12 \cdot 2^{2a_r}} \cdot k \tag{11}$$

C. Device Error Coupled with ADC Conversion

Due to the limitations of device intrinsic properties and process variation, NVM devices suffer from conductance variation, which results in a deviation between the target and real conductance. The non-ideal effects of NVM devices mainly include conductance reading variation [21], programming error [18], Stuck-At-Fault (SAF) [28], and thermal noise [29]. Among these, SAF defect is classified as a deterministic error, meaning the device is permanently stuck at either the low or high conductance state, called Stuck-At-Zero (SA0) and Stuck-At-One (SA1), respectively. In contrast, other device defects are considered stochastic errors, with the conductance variation represented by a Gaussian distribution [7], [23]:

$$g = g_{target} + g_{target} \cdot \mathcal{N}(0, \gamma^2), \tag{12}$$

where γ is a variation factor related to the device's intrinsic properties. The latter term is injected device error E_{dev} with variance $\sigma^2_{dev} = \gamma^2 \cdot \sigma^2_w$. Note that the device conductance value g_{target} is normalized. Then the variance of accumulated device error E_{dev_weight} in the weight dimension can be calculated as follows:

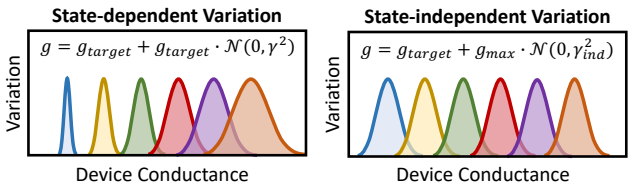

Fig. 6: Two types of conductance variation in NVM devices.

$$\sigma^2_{dev\_weight} = s^2 \cdot \sum_{i=0}^{b_w - 1} 2^{2i} \cdot \gamma^2 \cdot \sigma^2_{w_i} \tag{13}$$

Note that even if the conductance deviation of the actual device does not perfectly follow the above ideal Gaussian distribution, the derivation of (13) can still be performed with the calculated variance σ^2_{dev} of non-Gaussian distribution.

Although SAF defects are not recoverable, they can be modeled with a probability distribution in the device dimension. Assume that the probabilities of SA0 and SA1 are α_0 and α_1, respectively. The device conductance can be modeled as a three-component mixed distribution:

$$g = \begin{cases} 0 & P = \alpha_0, \\ 1 & P = \alpha_1, \\ g_{target} + g_{target} \cdot \mathcal{N}(0, \gamma^2) & P = 1 - \alpha_0 - \alpha_1. \end{cases} \tag{14}$$

Assuming the mean of the pretraining weight to be zero, the variance of the above mixed distribution σ^2_{dev} equals the weighted sum of each distribution's variance under the given probability:

$$\sigma^2_{dev} = \alpha_0 \cdot \alpha^2_w + \alpha_1 \cdot \alpha^2_w + (1 - \alpha_0 - \alpha_1) \cdot \gamma^2 \cdot \sigma^2_w. \tag{15}$$

Given that the effect of the off-state current caused by the device's finite on/off resistance ratio (R-ratio) can be perfectly eliminated with the aid of a *dummy column* [4], the R-ratio non-ideal factor is not included in (15). Besides, not all the devices exhibit state-dependent variation property shown in (12), in which the conductance variation is proportional to the conductance state [17]. As Fig. 6 shows, the expected variation in some devices does not depend on the specific conductance state and the real conductance can be expressed as: $g = g_{target} + g_{max} \cdot \mathcal{N}(0, \gamma^2_{ind})$, where γ_{ind} is the corresponding independent variation factor. In such a case, the variance of the injected device error σ^2_{dev} equals γ^2_{ind} and (15) needs to be modified to:

$$\sigma^2_{dev} = \alpha_0 \cdot \alpha^2_w + \alpha_1 \cdot \alpha^2_w + (1 - \alpha_0 - \alpha_1) \cdot \gamma^2_{ind}. \tag{16}$$

Notably, the actual device error is tightly coupled to the ADC conversion error, both influenced by the ADC conversion interval. Specifically, the device errors will accumulate with the current accumulation along the column direction and are digitized through ADCs. As shown in Fig. 7, the accumulated device error distribution $\mathcal{N}(0, S \cdot \sigma^2_{dev_i})$ is rounded to zero within the range of $[-C_{INR_i}/2, C_{INR_i}/2]$ and thus exhibits no impact on NN's inference accuracy. We propose an attenuation factor A to consider the effect of ADC conversion interval

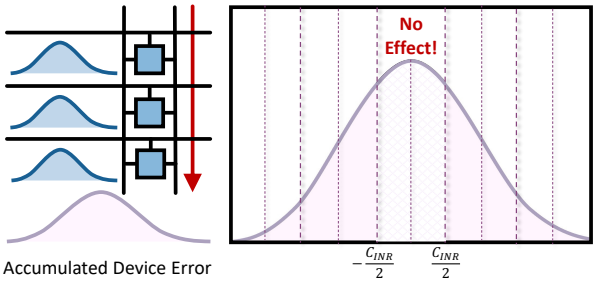

Fig. 7: Effect of ccumulated device error is attenuated by ADC conversion.

on device error, which is defined as:

$$A_i = 1 - \frac{\int_{-\frac{C_{INR_i}}{2}}^{\frac{C_{INR_i}}{2}} P(X=x)\cdot x^2 dx}{S\cdot\sigma^2_{dev_i}}, \tag{17}$$

where $P(X=x)$ is the probability density function (PDF) of $\mathcal{N}(0, S\cdot\sigma^2_{dev_i})$. The latter term represents the ratio of the variance within the range of $[-C_{INR_i}/2, C_{INR_i}/2]$ to the total variance $S\cdot\sigma^2_{dev_i}$. Finally, the calculation of (13) can be updated according to (15) and (17) to further consider the impact of SAF defect and ADC's coupling effect:

$$\sigma^2_{dev\_weight} = s^2 \cdot \sum_{i=0}^{b_w-1} 2^{2i}\cdot\sigma^2_{dev_i}\cdot A_i. \tag{18}$$

D. Unified Error

Given a specific set of software and hardware parameters, we can unify all errors into the weight dimension and obtain the variance of $E_{qua}, E_{adc\_weight}, E_{dev\_weight}$. With the coupling effect considered in the derivation, various errors exhibit independent impacts on accuracy. Then, the variance of unified error E in the weight dimension is obtained by:

$$\sigma^2_{err} = \sigma^2_{qua} + \sigma^2_{adc\_weight} + \sigma^2_{dev\_weight}. \tag{19}$$

Combining (4), (5), and (19), we obtain the non-ideal PIM accuracy (NIPA) evaluation model considering various coupling errors on analog PIM architectures:

$$NIPA = \frac{\sum_{i=0}^{b_w-1} 2^{2i}\sigma^2_{w_i}}{\frac{3\sigma^2_w}{s^2 2^{2b_w}} + \sum_{i=0}^{b_w-1} 2^{2i}\left(\frac{k\cdot S\cdot(1-sp_i)^2}{12\cdot 2^{2a_r}} + \sigma^2_{dev_i}A_i\right)} \tag{20}$$

Given a pre-trained neural network, we first collect NN's prior information. Then we calculate NIPA score based on (20) to evaluate the relative accuracy. Based on the NIPA evaluation model, the impact of various errors can be jointly analyzed without any evaluation overhead. This allows for a deeper exploration of different software and hardware parameters, facilitating further co-optimization of both algorithmic and hardware performance. A more detailed discussion is presented in Section VII.

VI. Non-Slicing Absolute Accuracy Evaluation

Based on the insight of unified error variance calculation, we further propose a non-slicing absolute accuracy evaluation method. As introduced in Section V, all the errors can be unified and calculated in the weight dimension. Therefore,

Algorithm 1 Non-Slicing Absolute Accuracy Evaluation

Input: Target model $\mathcal{F}$, pretrained weights $\mathbf{W}$, validation dataset $\mathbf{D}$, bit width b_w, ADC resolution a_r, crossbar size S, device conductance variation factor γ.
Output: Evaluated absolute accuracy A.

1: Partition $\mathbf{W}$ into N layers ($\mathbf{W}_{1\to N}$)
2: **for** $i = 1 \to N$ **do**
3: **# Prior information calculation**
4: σ^2_w = CalWeightVari($\mathbf{W_i}$, b_w, S)
5: s = CalAvgScalingFactor($\mathbf{W_i}$, b_w, S)
6: sp = CalAvgSparsity($\mathbf{W_i}$, b_w, S)
7: k = CalXbarNum($\mathbf{W_i}$, S)
8: **# Unified weight error injection**
9: σ^2_{err} = CalErrorVari(b_w, a_r, S, s, sp, k, γ)
10: $\mathbf{W}^*_\mathbf{i} = \mathbf{W_i} + \mathbf{W_i}\cdot\mathcal{N}(0, \frac{\sigma^2_{err}}{\sigma^2_w})$
11: **# Absolute accuracy evaluation**
12: $A = \mathcal{F}(\mathbf{D}|\mathbf{W}^*)$

we directly apply the unified weight error to the pre-trained model's weights, similar to (3):

$$\mathbf{W}^* = \mathbf{W} + \mathbf{W}\cdot\mathcal{N}(0, \frac{\sigma^2_{err}}{\sigma^2_w}), \tag{21}$$

where $\mathbf{W}$ and $\mathbf{W}^*$ are original and error-injected weights, respectively. The second term is the injected error, with a variance of σ^2_{err}. Different from the NIPA evaluation model, the calculation and injection of σ^2_{err} in the absolute accuracy evaluation part is performed layer by layer. The detailed evaluation process is shown in Algorithm 1. For each layer in a given model, we separately statistic its bit-level weight variance, sparsity, and other required information (Algorithm 1 **line 4∼7**). Then the unified error variance is calculated and the weight matrix of each layer is updated according to (21) (Algorithm 1 **line 9∼10**).

With this method, the unified errors are directly introduced in the weight dimension and the absolute accuracy on target dataset can be evaluated during the original algorithmic inference process (Algorithm 1 **line 12**), eliminating the need for a bit-and-crossbar slicing process.

VII. Experiments

A. Experimental Setup

Benchmarks. We evaluate the proposed accuracy evaluation methods on both CNNs and LLMs. ResNet-18 [1], MobileNet-v2 [30], and EfficientNet-v2m [31] are evaluated on CIFAR10 and CIFAR100 datasets. LLMs including Mamba-130M [32], RWKV-169M [33] and OPT-125M [34] are evaluated on PIQA [35] and Arc-Easy [36] datasets. Due to the intolerable overhead of slicing-based paradigm (i.e., several days per parameter set), experiments are not conducted on larger LLMs.

Methodology. In this work, the proposed accuracy evaluation methods are compared and verified with the absolute accuracy evaluated by DNN+NeuroSIM [4]. Adjustable parameters include weight bit-width b_w {4, 6, 8}, device conductance variation factor γ {0, 0.1, 0.2}, ADC resolution a_r {4, 5, 6, 7, 8} and crossbar size S {128, 256}. Specific parameter configuration for complex device behavior is listed in Fig. 11. Due to the

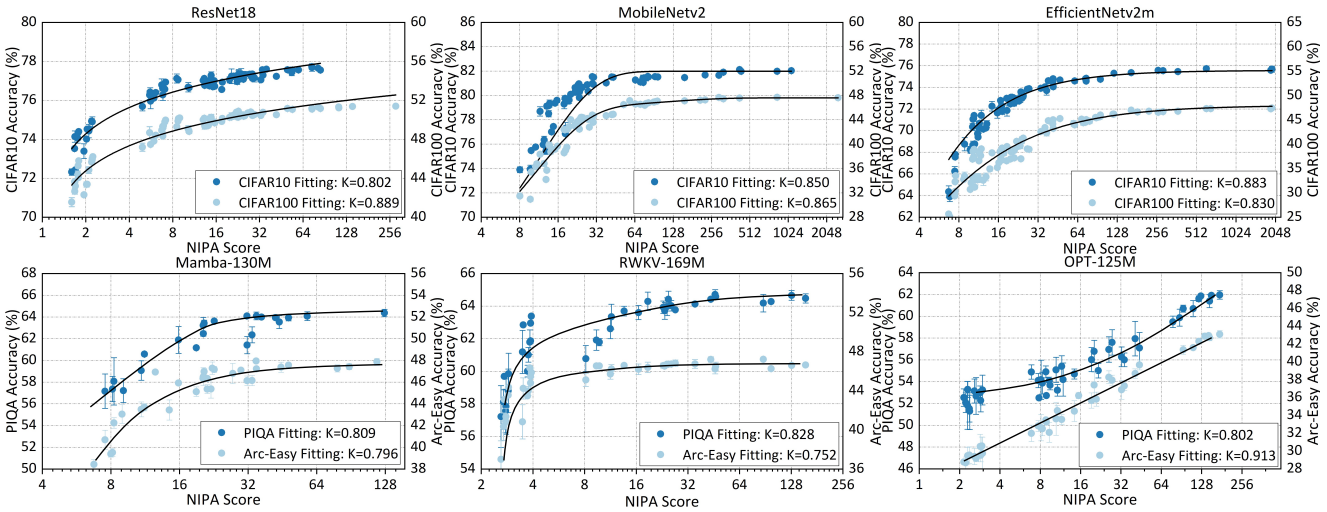

Fig. 8: High Kendall rank correlations K between NIPA score and ground truth accuracy provided by bit-and-crossbar slicing simulations. (Black line: the data points are automatically fitted using the nonlinear function)

TABLE II: Kendall rank correlation achieved by NIPA evaluation model for the evaluation of different LLM layers.

Model	Dataset	layer 1	layer 2	layer 3
OPT-125M	PIQA	0.802	0.789	0.740
	Arc Easy	0.913	0.734	0.774

randomness of device error, we conduct three experiments for each parameter set and take the average accuracy as the ground truth. We modify the MNSIM 2.0 simulator [5] based on the hardware performance model proposed in [37] to evaluate the PIM hardware performance. The proposed evaluation method is open-sourced on https://github.com/gld17/NIPA.git.

Metrics. To demonstrate the effectiveness of the proposed method, we report the Kendall ranking correlation and Mean Absolute Error (MAE) results for relative and absolute accuracy evaluation, respectively.

B. Performance of NIPA Evaluation Model

Fig. 8 presents the detailed results of NIPA score and ground truth accuracy for all possible software and hardware parameters. As shown in Fig. 8, the proposed NIPA evaluation model achieves high KD correlations consistently across both CNNs and LLMs, reaching up to 0.91. The smaller the NIPA score is, the more significantly NN accuracy is affected by the error (larger accuracy variation in three results of a same parameter set). We observe that ResNet18 and LLMs exhibit lower average NIPA scores compared to MobileNetv2 and Efficientnetv2m. This is because the weight matrix sizes of these models generally exceed the crossbar size (128/256), in which case, the ADC conversion errors will multiply and dominate. Due to the extremely long evaluation latency of LLMs, we only inject errors into their first block and evaluate the inference accuracy in an end-to-end manner. As the inference errors from the first block might be affected in subsequent propagation, which is difficult to account for, the Kendall rank correlations achieved by NIPA evaluation model on LLMs are slightly lower than those on CNNs. We further conduct additional experiments, injecting the errors into different layers separately. As shown in Table II, the proposed NIPA evaluation model maintains high Kendall rank correlations across different layers of OPT-125M.

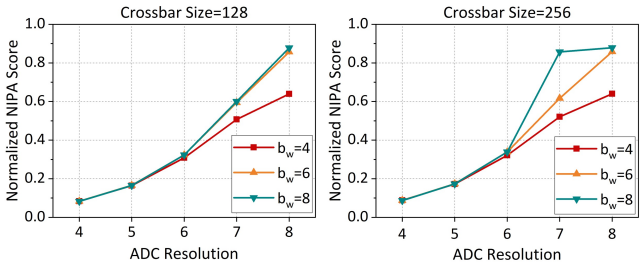

Fig. 9: The effect of different parameter sets on the ResNet-18 model (device errors are not considered).

Based on the NIPA evaluation model, we take ResNet-18 as an example and analyze the influence of different parameters on NN's inference accuracy. As shown in Fig. 9, ADC resolution exhibits a more significant effect on NN accuracy than other parameters. In the case of large ADC resolution (i.e., 7 or 8), increasing the weight bit-width can also provide significant performance benefits. However, when ADC resolution is set to less than 7, the change of weight bit-width shows a negligible effect on NN's accuracy. In this scenario, the ADC conversion error of each bit becomes relatively large. As a result, increasing the weight bit width will also amplify the ADC quantization error, which diminishes the performance gains from reducing the weight quantization error.

C. Effectiveness of NIPA on Complex Device Behavior

To validate the effectiveness of the proposed NIPA evaluation model on the devices with complex behavior, we further consider the devices with state-independent Gaussian-based and state-dependent Laplace-based conductance variation. With other software & hardware parameters fixed, adjustable device-related parameters are listed in the upper table of Fig. 11. The proposed NIPA evaluation model maintains a

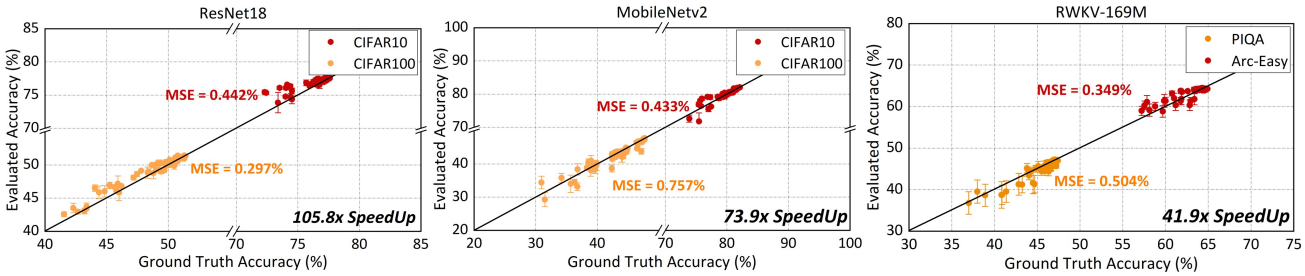

Fig. 10: Mean average error (MAE) between ground truth accuracy and non-slicing absolute accuracy evaluation results.

Parameter Configuration for Complex Device Behavior			
γ	{0, 0.1, 0.2, 0.3}	γ_{ind}	{0, 0.05, 0.1, 0.15}
l_b	{0, 0.06, 0.12, 0.18}	α	{1e-4, 1e-3, 1e-2}

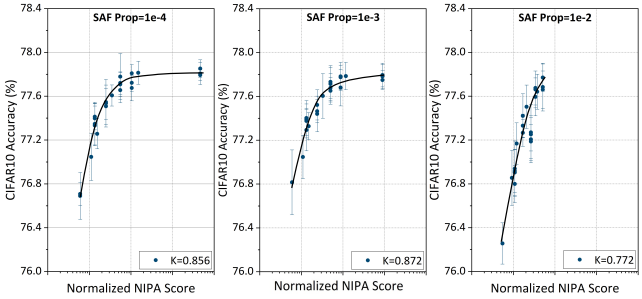

Fig. 11: NIPA evaluation model performs well for complex device behaviors on ResNet-18.

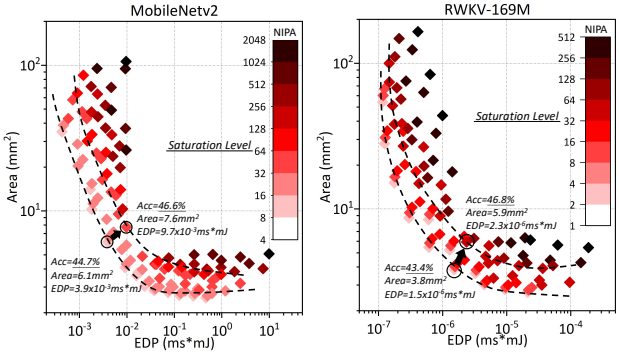

Fig. 12: NIPA evaluation model helps to achieve a better trade-off between algorithm and hardware performance.

high KD correlation of up to 0.913 on devices with complex behaviors. The NIPA evaluation model is not limited to the ideal Gaussian distribution of device conductance but depends only on the actual variance.

D. Non-Slicing Absolute Accuracy Evaluation Performance

To validate the effectiveness of the proposed non-slicing absolute accuracy evaluation method, we conduct three experiments for each parameter set and take the average accuracy as the evaluated accuracy. As Fig. 10 shows, low MAEs between ground truth and simulated accuracy down to 0.297% are achieved across different types of models and datasets with up to 105.8x speedup compared to DNN+NeuroSim.

E. Co-exploration Embedded with NIPA Evaluation Model

We further validate the effectiveness of the NIPA evaluation model within a PIM parameter search framework. Searching for the parameters on PIM architecture towards both algorithm and hardware performance requires PIM accuracy evaluations in the loop, incurring heavy runtime overhead. In our search framework, we incorporate the NIPA evaluation model to replace the bit-and-crossbar slicing-based evaluation. As shown in Fig. 12, when not considering algorithm performance under various errors, the NIPA scores of Pareto optimal parameters between PIM area and Energy-Delay Product (EDP) are the lowest, far below the accuracy saturation level (NIPA score of the parameter set with an accuracy close to that of the pre-trained model). Referring to Fig. 8, the inference accuracy of Mobilenetv2 and RWKV models approaches saturation when NIPA reaches 64 and 32, respectively. By setting the saturation level of the NIPA score as the target, we can search for a new Pareto curve that meets the NN inference accuracy requirements, achieving a better trade-off between algorithm and hardware performance.

VIII. Conclusion

In this work, we propose a non-ideal PIM accuracy (NIPA) evaluation model for relative accuracy evaluation and a non-slicing absolute accuracy evaluation method. By unifying various errors in the weight dimension and using NN's prior information, the proposed methods eliminate the need for the time-consuming bit-and-crossbar slicing simulation process while accounting for the coupling effects among various errors. Extensive experiments on CNNs and LLMs validate the effectiveness of the proposed methods. Furthermore, the joint search results also demonstrate that the NIPA evaluation model helps achieve a better trade-off between algorithm and hardware performance on PIM architectures.

Acknowledgment

This work was supported by the National Key R&D Program of China (2023YFB4502200), the National Natural Science Foundation of China (62325405, U24B601), Tsinghua University Initiative Scientific Research Program, Tsinghua-Meituan Joint Institute for Digital Life. Tsinghua-Efort Joint Research Center for EAI Computation and Perception, Beijing National Research Center for Information Science, Technology (BNR2024TD03001), Beijing Innovation Center for Future Chips, and State Key laboratory of Space Network and Communications. This research was partially supported by ACCESS–AI Chip Center for Emerging Smart Systems, sponsored by InnoHK funding, Hong Kong SAR (HKSAR) and Research Grants Council of HKSAR (16213824).

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[2] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[4] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, "Dnn+ neurosim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies," in *2019 IEEE international electron devices meeting (IEDM)*. IEEE, 2019, pp. 32–5.

[5] Z. Zhu, H. Sun, T. Xie, Y. Zhu, G. Dai, L. Xia, D. Niu, X. Chen, X. S. Hu, Y. Cao *et al.*, "Mnsim 2.0: A behavior-level modeling tool for processing-in-memory architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 11, pp. 4112–4125, 2023.

[6] M.-Y. Lin, H.-Y. Cheng, W.-T. Lin, T.-H. Yang, I.-C. Tseng, C.-L. Yang, H.-W. Hu, H.-S. Chang, H.-P. Li, and M.-F. Chang, "Dl-rsim: A simulation framework to enable reliable reram-based accelerators for deep learning," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2018, pp. 1–8.

[7] T. Shahroodi, G. Singh, M. Zahedi, H. Mao, J. Lindegger, C. Firtina, S. Wong, O. Mutlu, and S. Hamdioui, "Swordfish: a framework for evaluating deep neural network-based basecalling using computation-in-memory with non-ideal memristors," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, 2023, pp. 1437–1452.

[8] I. Chakraborty, M. F. Ali, D. E. Kim, A. Ankit, and K. Roy, "Geniex: A generalized approach to emulating non-ideality in memristive xbars using neural networks," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[9] Z. He, J. Lin, R. Ewetz, J.-S. Yuan, and D. Fan, "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*, 2019, pp. 1–6.

[10] C. Wang, Z. Chen, and S. Huang, "Micsim: A modular simulator for mixed-signal compute-in-memory based ai accelerator," *arXiv preprint arXiv:2409.14838*, 2024.

[11] L. Han, R. Pan, Z. Zhou, H. Lu, Y. Chen, H. Yang, P. Huang, G. Sun, X. Liu, and J. Kang, "Comn: Algorithm-hardware co-design platform for non-volatile memory based convolutional neural network accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[12] Z. Yan, X. S. Hu, and Y. Shi, "Computing-in-memory neural network accelerators for safety-critical systems: Can small device variations be disastrous?" in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, 2022, pp. 1–9.

[13] H. Sun, Z. Zhu, C. Wang, X. Ning, G. Dai, H. Yang, and Y. Wang, "Gibbon: An efficient co-exploration framework of nn model and processing-in-memory architecture," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 11, pp. 4075–4089, 2023.

[14] X.-J. Chen, C. Kuan, and C.-L. Yang, "Unified agile accuracy assessment in computing-in-memory neural accelerators by layerwise dynamical isometry," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.

[15] W. Zhang, X. Peng, H. Wu, B. Gao, H. He, Y. Zhang, S. Yu, and H. Qian, "Design guidelines of rram based neural-processing-unit: A joint device-circuit-algorithm analysis," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

[16] B. Wu, Y. Liu, J. Liu, H. Cheng, X. Wei, W. Tong, and D. Feng, "Fadesim: Enable fast and accurate design exploration for memristive accelerators considering non-idealities," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[17] T. P. Xiao, B. Feinberg, C. H. Bennett, V. Prabhakar, P. Saxena, V. Agrawal, S. Agarwal, and M. J. Marinella, "On the accuracy of analog neural network inference accelerators," *IEEE Circuits and Systems Magazine*, vol. 22, no. 4, pp. 26–48, 2022.

[18] G. L. Zhang, B. Li, X. Huang, C. Shen, S. Zhang, F. Burcea, H. Graeb, T.-Y. Ho, H. Li, and U. Schlichtmann, "An efficient programming framework for memristor-based neuromorphic computing," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 1068–1073.

[19] W. He, J. Meng, S. K. Gonugondla, S. Yu, N. R. Shanbhag, and J.-s. Seo, "Prive: efficient rram programming with chip verification for rram-based in-memory computing acceleration," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2023, pp. 1–6.

[20] L. Chen, J. Li, Y. Chen, Q. Deng, J. Shen, X. Liang, and L. Jiang, "Accelerator-friendly neural-network training: Learning variations and defects in rram crossbar," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 19–24.

[21] Z. Yan, Y. Qin, W. Wen, X. S. Hu, and Y. Shi, "Improving realistic worst-case performance of nvcim dnn accelerators through training with right-censored gaussian noise," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 1–9.

[22] Y. Lim, D. Kim, and J. Kim, "Selcc: Enhancing mlc reliability and endurance with single-cell error correction codes," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2024, pp. 1–6.

[23] S. Jain, A. Sengupta, K. Roy, and A. Raghunathan, "Rxnn: A framework for evaluating deep neural networks on resistive crossbars," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 2, pp. 326–338, 2020.

[24] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 27–39, 2016.

[25] A. Bhattacharjee, L. Bhatnagar, and P. Panda, "Examining and mitigating the impact of crossbar non-idealities for accurate implementation of sparse deep neural networks," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 1119–1122.

[26] A. Bhattacharjee, Y. Kim, A. Moitra, and P. Panda, "Examining the robustness of spiking neural networks on non-ideal memristive crossbars," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, 2022, pp. 1–6.

[27] T. Cao, C. Liu, W. Wang, T. Zhang, H. K. Lee, M. H. Li, W. Song, Z. X. Chen, V. Y.-Q. Zhuo, N. Wang *et al.*, "A non-idealities aware software–hardware co-design framework for edge-ai deep neural network implemented on memristive crossbar," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 4, pp. 934–943, 2022.

[28] L. Xia, W. Huangfu, T. Tang, X. Yin, K. Chakrabarty, Y. Xie, Y. Wang, and H. Yang, "Stuck-at fault tolerance in rram computing systems," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 1, pp. 102–115, 2017.

[29] L. Kish and C. Granqvist, "Noise in nanotechnology," *Microelectronics Reliability*, vol. 40, no. 11, pp. 1833–1837, 2000.

[30] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[31] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *International conference on machine learning*. PMLR, 2021, pp. 10 096–10 106.

[32] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.

[33] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, S. Biderman, H. Cao, X. Cheng, M. Chung, M. Grella *et al.*, "Rwkv: Reinventing rnns for the transformer era," *arXiv preprint arXiv:2305.13048*, 2023.

[34] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin *et al.*, "Opt: Open pre-trained transformer language models," *arXiv preprint arXiv:2205.01068*, 2022.

[35] Y. Bisk, R. Zellers, J. Gao, Y. Choi *et al.*, "Piqa: Reasoning about physical commonsense in natural language," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 05, 2020, pp. 7432–7439.

[36] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, "Think you have solved question answering? try arc, the ai2 reasoning challenge," *arXiv preprint arXiv:1803.05457*, 2018.

[37] J. Sun, P. Houshmand, and M. Verhelst, "Analog or digital in-memory computing? benchmarking through quantitative modeling," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 1–9.