Reliability-Aware Training and Performance Modeling for Processing-In-Memory Systems

Hanbo Sun*, Zhenhua Zhu*, Yi Cai*, Shulin Zeng, Kaizhong Qiu, Yu Wang, Huazhong Yang
Dept. of EE, BNRist, Tsinghua University
yu-wang@tsinghua.edu.cn

ABSTRACT

Memristor based Processing-In-Memory (PIM) systems give alternative solutions to boost the computing energy efficiency of Convolutional Neural Network (CNN) based algorithms. However, Analog-to-Digital Converters' (ADCs) high interface costs and the limited size of the memristor crossbars make it challenging to map CNN models onto PIM systems with both high accuracy and high energy efficiency. Besides, it takes a long time to simulate the performance of large-scale PIM systems, resulting in unacceptable development time for the PIM system. To address these problems, we propose a reliability-aware training framework and a behavior-level modeling tool (MNSIM 2.0) for PIM accelerators. The proposed reliability-aware training framework, containing network splitting/merging analysis and a PIM-based non-uniform activation quantization scheme, can improve the energy efficiency by reducing the ADC resolution requirements in memristor crossbars. Moreover, MNSIM 2.0 provides a general modeling method for PIM architecture design and computation data flow; it can evaluate both accuracy and hardware performance within a short time. Experiments based on MNSIM 2.0 show that the reliability-aware training framework can improve 3.4× energy efficiency of PIM accelerators with little accuracy loss. The equivalent energy efficiency is 9.02 TOPS/W, nearly 2.6~4.2× compared with the existing work. We also evaluate more case studies of MNSIM 2.0, which help us balance the trade-off between accuracy and hardware performance.

1 INTRODUCTION

In recent years, Convolutional Neural Network (CNN) based algorithms have made breakthrough improvements in various fields. However, besides the high accuracy, the CNN-based algorithms suffer from a massive amount of network parameters and computations caused by the complex network structure, making CNN-based algorithms consume high energy and a long computation time.

Existing researches show that compared with ASIC and GPU solutions, the emerging memristors (*e.g.*, RRAM, Resistive Random Access Memory) and memristor-based Processing-In-Memory (PIM) architecture can improve the computing performance and energy efficiency of CNN by more than 100× [7]. The PIM system's performance improvement comes from the memristor crossbar structure, which can perform Matrix Vector Multiplications (MVMs) in memory. When applying the input voltage vector onto the word-lines (WLs) of the memristor crossbar and mapping the weight matrix to the memristor conductance, we can obtain the MVM results from the current/voltage on the bit-lines (BLs) by using Analog-to-Digital Converters (ADCs). Therefore, the PIM system can eliminate the weight data movements between memory and computing units in traditional von Neumann architectures, which are also the most time- and energy-consuming part of CNN models.

However, utilizing PIM systems to accelerate CNN models is still challenging. On the one hand, on account of the high interface costs of ADCs and the limited size of the memristor crossbars, directly mapping the CNN models onto the PIM system will cause undesirable energy consumption and accuracy loss for three reasons. Firstly, PIM accelerators based on high-resolution ADCs can achieve low accuracy loss even in large scale datasets and complicated CNN models. However, high-resolution ADCs cost more energy than low-resolution ADCs (e.g., for one conversion, the 8-bit ADC in [5] consumes 8.66× more energy than the 4-bit ADC in [24]). Since over 60% energy of the entire PIM accelerators is consumed by the ADCs [33], high-resolution ADCs seriously ruin the system's overall energy efficiency. Secondly, to improve the overall energy efficiency, existing studies have proposed several PIM architectures with low-resolution ADCs. However, these architectures have undesirable accuracy loss in large scale datasets or complex CNN models (*e.g.*, Utilizing 4-bit ADCs results in 8% accuracy loss on ResNet18 @ cifar10 [3]). Thirdly, On account of the immature technology and non-ideal factors (*e.g.*, IR-drop), the memristor crossbar can only be manufactured with limited size (*e.g.*, 128×128). And the limited size makes it inevitable to split the large weight matrix into multiple memristor crossbars, resulting in extra energy consumption and accuracy loss. Therefore, how to reduce the ADCs' resolution further, while ensuring the computing accuracy within limited memristor crossbar size, is of great importance for improving the overall energy efficiency of the PIM accelerators.

On the other hand, the vast architecture search space and the complicated CNN models in the PIM accelerators cause a long SPICE simulation time, resulting in an unacceptable long development time for the iteration of design, simulation, and updating [31]. To tackle this problem, existing studies have proposed several behavior-level simulators or modeling frameworks to evaluate the accuracy or energy efficiency within a short time [12, 22, 31]. MNSIM [31] is a behavior-level simulator, which provides a hierarchical hardware structure abstraction to deploy different CNN models. Although the crossbars' data flow can be quickly evaluated in MNSIM, it lacks simulation of other digital parts in PIM accelerators. Besides, the accuracy simulation of MNSIM is only applicable to MVMs of a single crossbar other than the entire CNN model. NeuroSim [25] is a circuit-level simulator which provides detailed device- and circuit-level evaluations, but it lacks flexibility in supporting various CNN structures at the algorithm level. DL-RSIM [22] and PytorX [12] are PIM simulators to evaluate the accuracy of the target CNN

*: All authors contributed equally to this work.

ASPDAC '21, January 18–21, 2021, Tokyo, Japan

ACM ISBN 978-1-4503-7999-1/21/01…$15.00
https://doi.org/10.1145/3394885.3431633

models considering device non-ideal factors. However, hardware simulation is missed in these tools.

To fully exploit the energy efficiency and performance of PIM-based CNN accelerators, we propose a reliability-aware training framework and a behavior-level modeling tool (MNSIM 2.0) for PIM accelerators. For the former one, the proposed reliability-aware training framework contains the network splitting/merging analysis and a PIM-based non-uniform activation quantization scheme, which can reduce the ADC resolution requirements in splitting crossbars and improve the energy efficiency of the PIM systems. For the latter one, to evaluate both hardware performance and test accuracy within a short time, the proposed behavior-level modeling framework designs the base architecture and constructs a hardware performance modeling flow.

The main contributions of this paper are listed as follows:

(1) We propose a reliability-aware training framework, containing network splitting/merging analysis and a PIM-based non-uniform activation quantization scheme. By reducing the ADC resolution by 2 bits, the proposed training framework achieves 70% energy consumption reduction and comparable accuracy to traditional activation quantization schemes.
(2) We propose MNSIM 2.0, a behavior-level modeling framework that can simulate the accuracy of CNN models considering data splitting strategies and ADC resolutions, and generate the corresponding hardware performance.
(3) Experiments based on MNSIM 2.0 show that the reliability-aware training framework can improve 3.4× energy efficiency of PIM accelerators with little accuracy loss. Besides, more case studies of MNSIM 2.0 are evaluated, which help us to balance the trade-off between accuracy and hardware performance.

2 PRELIMINARY

2.1 Convolutional Neural Network

The convolutional (CONV) layer is the main component of a typical CNN model. CONV layers perform the convolution operation, which is shown as:

$$F_o(x,y,z)=\sigma\Big(\sum_{i=0}^{K-1}\sum_{j=0}^{K-1}\sum_{k=1}^{C_{in}} F_i(x+i,y+j,k)w_z(i,j,k)\Big) \tag{1}$$

where F_i and F_o denote the 3-dimensional input and output data, respectively. w_z is the z^{th} 3-dimensional convolution weights. K and C_{in} represent the kernel size and the input channel of the z^{th} weights, respectively. By multiplying the corresponding elements and then summing, we can get the intermediate result in (x, y, z). Then, we use $\sigma(\cdot)$, a nonlinear activation function, to transfer the intermediate result to the final output activation $F_o(x, y, z)$.

2.2 Memristor Basics

Memristors can store information by changing the resistance values and have the advantage of non-volatility. By combining memristors in the form of crossbar structure, we can achieve high-density storage and *in-situ* computing (*e.g.*, MVM, CONV) together. To realize efficient MVMs, we represent the input vector by voltages $\boldsymbol{V}$ applied onto the WLs of crossbars, and leverage the conductance of memristor to store the weight matrix. Then the MVM results can be derived by the output current vector $\boldsymbol{I}$ on BLs:

$$i_{out,k}=\sum_{j=1}^{N} g_{k,j}v_{in,j} \tag{2}$$

where $v_{in,j}$, $i_{out,k}$ denote the j^{th} element of the input voltage vector $\boldsymbol{V}$ and the k^{th} element of the output current vector $\boldsymbol{I}$; $g_{k,j}$ is the conductance of the (i, j) cell in the crossbar. Due to the MVMs are computed in the analog domain, some interfaces to transfer information between the analog domain and the digital domain (*e.g.*, Analog-to-Digital Converters (ADCs), Digital-to-Analog Converters (DACs)) are indispensable in the PIM system.

Benefit from the *in-situ* MVM computing pattern, the memristor crossbar-based PIM systems can achieve high energy efficiency by eliminating the matrix data movements.

3 COMPUTING RELIABILITY OF PROCESSING-IN-MEMORY SYSTEMS

Although high energy efficiency can be obtained in PIM systems, various non-ideal factors cause computation deviation and make the PIM-based CNN computing system inaccurate and unreliable. These non-ideal factors include two aspects: circuit-level non-ideal factors and device faults.

The circuit-level non-ideal factors can be divided into two categories, *i.e.*, the impact of interconnections [9] and the quantization error caused by DACs/ADCs [33]. As the technology node scales down, the parasitic parameters caused by interconnections show negative impact on computing accuracy. To be specific, wire resistance, IR drop, and sneak path will disturb the value of stored weight (*i.e.*, conductance), input activation (*i.e.*, input voltage), and MVMs' results (*i.e.*, bit-line current), respectively. Existing work adapts circuits optimization [10], technological exploration [9], and circuits modeling and network retraining [12] to tackle these problems. Since memristor perform MVMs in the analog domain, DACs and ADCs are needed for data conversion. Researchers have analyzed the effect of quantization error on calculation accuracy and energy consumption [33]. However, there still lacks interface optimization methods to optimize the interface costs while reducing the accuracy loss for complex data sets and algorithm models.

Because of the immature technologies, memristor device faults may occur during both chip fabrication (static faults) and memory read/write (dynamic faults). The device faults can be further classified as soft faults and hard faults. For the soft faults, the resistance of memristor is still changeable, while the actual resistance is not equal to the expected one. The typical soft faults include write variation [32], read disturbance [20], and non-linear resistance distribution [19]. For the hard faults, the resistance of memristor cannot be tuned. This category includes stuck-at-0 (SA0) and stuck-at-1 (SA1) faults on account of limited endurance [1] and fabrication defect [4]. Researchers have proposed series of fault tolerance methods for device faults to rescue the computation accuracy, containing resistance calibration [20], weight remapping [13, 30], and algorithm-hardware co-optimization [2, 6, 30].

In this paper, to implement reliable and energy efficient PIM systems, we focus on optimizing the interface-level non-ideal factors (*i.e.*, ADCs' quantization error) and provide a PIM system performance and accuracy evaluation tool considering different non-ideal factors for efficient design space exploration.

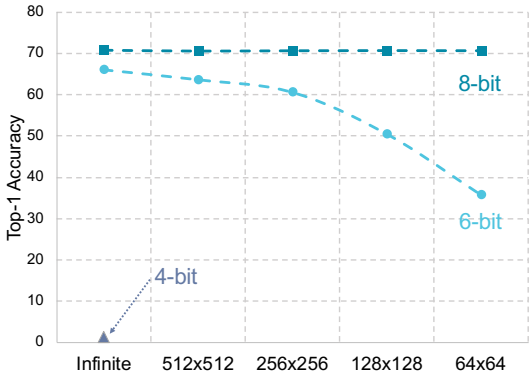

Figure 1: Accuracy of VGG16 on ImageNet dataset under different crossbar size and ADC resolution configurations.

4 RELIABLE-AWARE TRAINING FRAMEWORK

Many hardware parameters have crucial impacts on accuracy and energy efficiency of PIM systems, *e.g.*, interface resolution and crossbar size. However, existing training frameworks designed for CMOS-based accelerators ignore these parameters' influence, resulting in undesirable accuracy loss when mapping well-trained CNN models onto PIM systems. To tackle this problem, we propose a reliable-aware training framework for PIM systems, which can achieve negligible accuracy loss while further improving the energy efficiency of PIM systems. The proposed training framework utilizes weight matrix splitting, quantization range optimization, high-precision scale implementation, and non-uniform quantization to integrate hardware parameters into the CNN training.

4.1 Splitting & Quantization Analysis

On account of the limited crossbar size, we split and allocate the large convolutional layer in CNN models into multiple crossbars. Correspondingly, the convolutional operations are also divided into multiple sub-MVM blocks. In this scenario, we get the sub-MVM results from each crossbar and accumulate them through the adder tree to obtain the final result, as shown in Figure 5. However, this computational data flow introduces two sources of calculation deviation. On the one hand, limited by the resolution of ADCs, the high precision analog MVM results on BLs will be quantized, bringing quantization error. On the other hand, the accumulation operation expands the bit width of the intermediate data, and the final results need to be clipped. Existing training frameworks ignore the influence of these deviation sources, resulting in undesirable accuracy loss. As shown in Figure 1, the test accuracy substantially drops when directly mapping the models onto the size-limited crossbars with low ADC resolution. In our training framework, we model these calculation deviation sources and integrate them into the CNN training flow, which are shown as follows:

$$F(w,x) = Clip(\sum_{i=1}^{N} Quan(F(w_i, x_i))) \tag{3}$$

where $Clip(\cdot)$ and $Quan(\cdot)$ denote the clipping and quantization function, respectively. w_i and x_i means the weights and input in the i^{th} sub-MVM block, and N is the number of sub-MVM blocks.

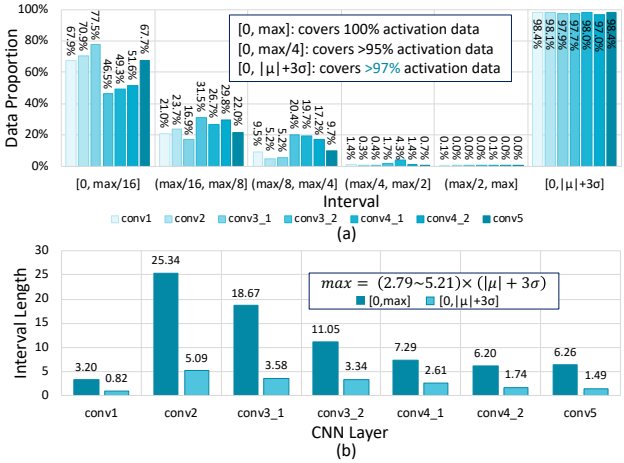

Figure 2: (a) The percentage of activation in different ranges w.r.t. different CNN layers in a well-trained VGG-8 model. (b) The quantization range sizes in different layers [27].

4.2 Quantization Range Optimization

In traditional quantization schemes, we utilize the maximum value of the input data as the quantization range. Although this broad quantization range can cover all the input data, the maximum value is easily affected by occasional extreme data and can not reflect the overall data distribution. As shown in Figure 2(a), the interval $[0, max/4]$ covers more than 95% activation data, meaning most of the data does not use the highest two bits. Referring to [21], the weights and the activation data in CNN models can be described by a Gaussian distribution. Therefore, we propose a quantization range optimization method to generate appropriate quantization range based on 3σ principle, which can cover more than 99.7% normal data appears in $[\mu - 3\sigma, \mu + 3\sigma]$, as shown below:

$$\alpha \leftarrow |mean(v_{in})| + 3std(v_{in}) \tag{4}$$

where v_{in} denotes the input data, and α is the quantization range. As shown in Figure 2, the new quantization range can cover more than 97% data, while the quantization range size is reduced to 25% after using the optimized quantization range. Simultaneously, by truncating the data beyond the quantization range for training, we can also guarantee the training accuracy in this quantization range.

4.3 High-precision Scale Implementation

Limited by the binary system in the digital circuit, the scaling factor in the traditional quantization method must be an integer power of 2. However, this constraint leads to a mismatch between the scaling factor and the quantization range, which further damages the quantization precision and leads to an increase in the quantization errors. To solve this problem, we propose a high-precision scale implementation scheme based on PIM. The proposed method utilizes memristor with adjustable resistance instead of resistor as the load resistance. Therefore, we can scale and map different ranges of currents to one fixed voltage range by adjusting the load resistance, which is equivalent to achieving high precision scaling factors.

Besides, in a typical CNN training phase, only part of the data is trained each time due to the storage limitation. There are different data distributions among different data, resulting in various scaling factors and non-convergence during training. To tackle this problem, we introduce the momentum smoothing method to balance

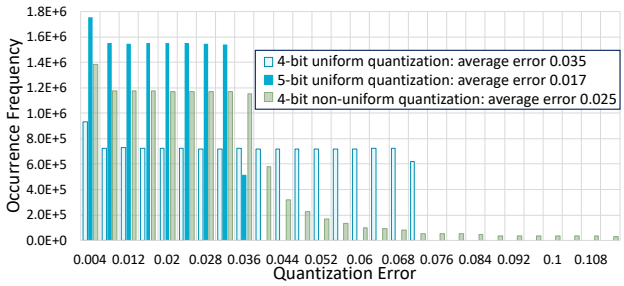

Figure 3: Quantization error under different quantization methods (i.e., 4-bit uniform quantization, 5-bit uniform quantization, and 4-bit non-uniform quantization) [27].

different data distributions and accelerate training convergence. After a new quantization scaling parameter is generated, it is weighted summed with the previous scaling parameter. Then we can get a smoothed quantization scaling parameter, as shown in Equation 5 (m denotes the momentum coefficient).

$$\alpha \leftarrow m\alpha + (1-m)\left(|mean(v_{in})| + 3std(v_{in})\right) \tag{5}$$

4.4 Non-uniform Quantization

As mentioned before, the activation data in CNN models is close to the Gaussian distribution. Therefore, when using traditional uniform quantization schemes to quantify the activation data, fixed quantization step brings large quantization error on the data-intensive interval. A straightforward solution is to adjust the quantization step (*i.e.*, non-uniform quantization) according to the data density. As shown in Figure 3, under the same quantization bits, non-uniform quantization can achieve smaller quantization error. Inspired by this observation, we realize the non-uniform quantization by non-uniform ADCs in our quantization scheme. At the circuit level, by adjusting the capacitance and the divider resistance value in ADCs, we can generate non-uniform reference voltage levels and realize the non-uniform quantization with little area and energy overhead. Correspondingly, at the algorithm level, to achieve non-uniform quantization, we utilize nonlinear functions (*e.g.*, Sigmoid) to determine the quantization step. After that, we use multiplexers to transfer low-precision non-uniform quantization results to high-precision uniform quantization data. Combining these two parts, our method can be expressed as:

$$\begin{aligned}\hat{x}_{q,out} &= \hat{Q}_k\left(x_{in}, \alpha\right) \\ &= Q_{2k}\left(f^{-1}\left(\frac{clip\left(r\left(f\left(\eta\frac{x_{in}}{\alpha}\right)2^k\right), 1, 2^k-1\right)}{2^k}\right)\frac{\alpha}{\eta}, \alpha\right)\end{aligned} \tag{6}$$

where f, r, and Q denote the *sigmoid*, *round*, and uniform quantization function, respectively. η is a hyperparameter used to adjust the distribution.

5 HARDWARE PERFORMANCE MODELING

In addition to the impact on accuracy, various hardware design parameters (*e.g.*, crossbar size, ADC resolution, etc.) also affect hardware area, performance, and energy efficiency. However, the huge design space and large-scale PIM circuits make the SPICE simulation time unacceptable. For the purpose of efficient design space exploration and CNN accuracy evaluation, we propose MNSIM 2.0 to model the performance of PIM systems. In the entire design flow, users provide CNN models and basic architecture design as inputs.

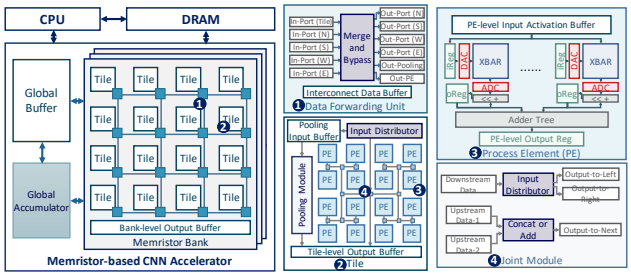

Figure 4: Left: The architecture design used for PIM systems; Right: Details of the Data Forwarding Unit, the Memristor Tile, the Process Element (PE), and the Joint Module [34].

Then, MNSIM 2.0 generates the hardware modeling results and CNN computing accuracy, which can be used to guide users to adjust the architecture design parameters. Once the hardware parameters are determined, the reliable-aware training framework can be utilized to optimize the CNN accuracy deployed on PIM systems.

5.1 Hierarchical Structure for PIM

In order to support complicated CNN models and describe different PIM architectures, we propose a hierarchical modeling structure and a basic architecture design, as shown in Figure 4. At the system level, the proposed architecture consists of the host CPU, off-chip DRAM, and a memristor-based CNN accelerator. The host CPU controls the weight mapping in the CNN deployment phase and wakes up the accelerator in the compute phase. At runtime, the accelerator reads input data from DRAM and uses the on-chip buffer to store all the intermediate data. After the accelerator completes the inference calculation, the result will be written back to DRAM. Inside the memristor-based CNN accelerator, there exist a global buffer, a global accumulator, and several memristor banks. The global buffer and global accumulator store the intermediate data and wait for other data to calculate the element summation layer to support the bypass network structure. From high-level to low-level, the hierarchical structure consists of memristor banks, memristor tiles, processing elements (PEs), and memristor crossbars. For each memristor bank, we organize a memristor tile array and connect them in a similar way to Network-on-Chip (NoC). In this connection scheme, each memristor tile is adjacent to the data forwarding unit, which receives data from other tiles, merges the intermediate data, and outputs the result to the local tile or other tiles. And inside the memristor tile, several PEs are connected as an H-tree structure to merge the result of this tile in joint modules.

5.2 Data Flow Construction

Based on the basic architecture design, we analyze the data flow of each memristor bank, as shown in Figure 5. The entire data flow consists of three parts:

1. Input data flow (the cyan-blue arrow): we get the input data from the data forwarding unit and write the data into the PEs' activation buffers through the H-tree.

2. Computation data flow (the violet arrow): we get the input data from the activation buffers and send the data to the input registers (iRegs). Considering the limited DAC resolution and WL/BL parallelism, we adopt multiple cycles to complete the whole computation of the input activation, while the crossbar needs to keep activated and the iRegs need to remain unchanged.

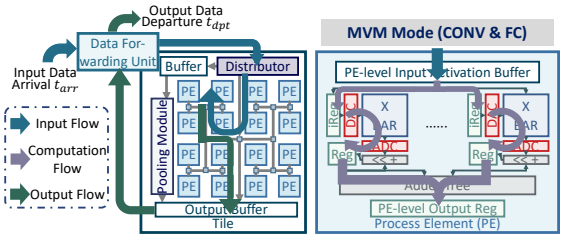

Figure 5: The data path of the memristor tile [34].

3. Output data flow (the green arrow): The output data flow is similar to the input data flow. We adopt the joint module to merge the results of PEs firstly, then send the merged PE results to the output buffer. For the next computation, we utilize data forwarding units to transfer data in the output buffer to other tiles.

5.3 Hardware Performance Estimation

Based on the architecture and data flow construction, the proposed modeling tool, MNSIM 2.0, estimates hardware performance in the following five levels, *i.e.*, memristor crossbar, analog and digital conversion interface, digital circuit, buffer, and NoC.

Memristor crossbar: For crossbar modeling, users can provide information from three aspects, *i.e.*, crossbar level, device level, and technology nodes. Here we use area estimation as an example to illustrate the crossbar modeling. MNSIM 2.0 gives four area estimation methods from different levels:

$$Xbar\_Area = \begin{cases} User's\ Given\ Value & \\ xbar_c \times xbar_r \times Device\_Area & \\ xbar_c \times xbar_r \times 3(W/L+1)F^2 & 1T1R \\ xbar_c \times xbar_r \times 4F^2 & 0T1R \end{cases} \quad (7)$$

$1T1R$ and $0T1R$ represent MOSFET-accessed and cross-point structure, W/L is the transistor technology parameter, F is the memristor technology node, and $xbar_c$ and $xbar_r$ denote the number of columns and rows of the crossbar, respectively.

Analog and digital conversion interface: In MNSIM 2.0, we count the conversion times of ADCs and DACs and give the analog and digital conversion interface estimation results based on the hardware parameters of ADCs and DACs. Like crossbars, users can provide the specific hardware parameters to configure ADCs and DACs. Meanwhile, MNSIM 2.0 also provides some latest DAC and ADC designs with different resolutions as default configurations.

Digital circuit: Besides the MVM part, digital modules are also essential for CNN models in the PIM system. Figure 4 presents some common digital modules in our PIM architecture, such as address decoders, H-trees, pooling modules, data forwarding units, etc. To estimate the performance of these digital modules, MNSIM 2.0 utilizes Synopsys Design Compiler® to synthesize digital circuit modules at TSMC 65*nm* technology node. And for digital circuit modules in other technology nodes, we use the scaling down estimation method to generate simulation results from 65*nm* results.

Buffer: As shown in Figure 4, we utilize on-chip buffers to store the input activation, interconnect data, tile-level output, and so on. In MNSIM 2.0, we use CACTI [28] and the method of fitting and looking up table to estimate read and write overhead of the buffers, based on user-specified buffer configuration.

NoC: In our PIM architecture, the NoC structure is used to merge the results of sub-MVM blocks and transfer the intermediate data to the next layer. In MNSIM 2.0, we use Manhattan distance to describe

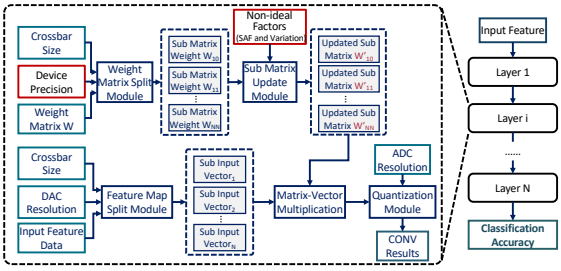

Figure 6: Accuracy evaluation of PIM-based CNN [34].

these two parts and estimate the NoC latency. We estimate the power and area of the NoC according to [16, 23].

5.4 Accuracy Estimation

In addition to the hardware performance estimation, accuracy estimation is also one crucial part of our simulator. As shown in Figure 6, we realize the accuracy evaluation by integrating hardware parameters and non-ideal factors into the calculation process of each layer based on the PyTorch framework. Firstly, considering the size of the crossbars, memristor precision, and DAC resolution, we divide the weight matrices and the input feature data into sub-matrices and sub-vectors, respectively. Secondly, to analyze the impact of non-ideal factors on the accuracy, we update values in each according to the effects of the non-ideal factors. Up to now, MNSIM 2.0 supports two non-ideal factors of the memristor, *i.e.*, stuck at fault (SAF) and resistance variations, which is described in Section 3, and more non-ideal factors will be supported in the future. Thirdly, we perform MVM in the updated sub-matrix to get intermediate results and quantize the intermediate results according to the ADC resolution and the user-specified quantization scheme. Finally, we accumulate the quantized intermediate results to obtain the final output of this layer, which is mentioned in Section 4.1. Performing these steps layer by layer, we can obtain the final accuracy simulation results in a short time.

6 EXPERIMENTS

6.1 Experiment Setup

To evaluate the proposed training framework and our simulator, we train several typical CNN models, *i.e.*, LeNet [18], modified VGG-8 [15], and ResNet [11], on the Cifar-10 dataset [14] as our benchmark. As for the hardware parameters used in our simulator, we refer to [29] to generate the memristors, and configure the power, area, latency, and resolution of the ADCs and DACs based on [5, 8, 17, 24, 26]. As for the digital parts, we model them at TSMC 65*nm* technology node with frequency of 500MHz.

6.2 Performance Comparison Using MNSIM 2.0

To evaluate the influence of ADC resolution, we use MNSIM 2.0 to simulate the system performance under different ADC configurations. The CNN accuracy and hardware performance results are shown in Table 1. The results demonstrate that the CNN computing power drops dramatically when the ADC resolution becomes lower. However, ADCs with lower resolution also bring higher quantization error, which will destroy the CNN function. Besides, experiment results show that different network models have different tolerance to the quantization error. Therefore, architecture designers should analyze the trade-off and select the appropriate

Table 1: Latency (L,ms), power (P,W), and accuracy (A,%) under different ADC resolutions (R) using single bit memristor devices. B is the baseline [34]

	LeNet			VGG-8			ResNet-20		
R	L	P	A	L	P	A	L	P	A
B	-	-	76.27	-	-	91.64	-	-	91.18
4	0.26	0.34	10.81	10.45	9.67	10.81	2.82	11.97	10.81
6	0.28	0.54	49.45	11.56	15.44	53.91	3.06	19.33	10.81
8	0.29	1.55	72.72	12.32	43.67	90.18	3.22	55.32	87.81
10	0.28	2.62	76.63	12.29	73.76	91.45	3.20	93.68	88.91

Table 2: Accuracy and energy consumption under different situations using 4-bit memristor devices (A and W are the precision of activations and weights) [27]

		LeNet		VGG-8-BN		ResNet-18	
		Accuracy	Energy/uJ	Accuracy	Energy/uJ	Accuracy	Energy/uJ
float baseline		0.7627	-	0.9336	-	0.8887	-
[3]	A8W4	0.7499	1.68	0.9308	2115.9	0.8785	19.8
	A6W4	0.7463	0.41	0.9243	458.2	0.8756	4.52
	A4W4	0.6375	0.15	0.1887	140.9	0.7412	1.56
Ours	A4W4	0.7467	**0.14**	0.9286	**138.8**	0.8655	**1.5**

ADC resolution according to the algorithm models and hardware performance constraints concurrently.

6.3 Training Framework Performance

Table 2 shows the performance of the proposed reliability-aware training framework compared with existing work [3], which reduces energy consumption of ADCs while keeping the accuracy. As shown in the last two rows, our training framework can achieve ~10% improvement on the test accuracy with the same quantization constraints on weights and activation. Besides, while achieving a comparable accuracy, the proposed training framework can reduce the ADC resolution by 2 bits and reduce ~70% energy consumption (equivalent to 3.4× energy efficiency improvement). It should be noted that the energy efficiency is also improved by the energy-aware weight regularization method introduced in [27].

7 CONCLUSION AND FUTURE WORK

In this paper, we propose a reliability-aware training framework and a behavior-level modeling framework (MNSIM 2.0) for PIM accelerators. The training framework contains network splitting/merging analysis and a PIM-based non-uniform activation quantization scheme to reduce the interface costs. And MNSIM 2.0 realizes the hardware performance modeling within a short time. Experiments based on MNSIM 2.0 show that the reliability-aware training framework can improve 3.4× energy efficiency of PIM accelerators with little accuracy loss. The equivalent energy efficiency is 9.02 TOPS/W, nearly 2.6~4.2× compared with existing work. In the future, we will integrate MNSIM 2.0 with other circuit-level simulators to achieve more accurate simulation results and support more hardware parameters.

8 ACKNOWLEDGEMENTS

This work was supported by National Key Research and Development Program of China (No. 2017YFA0207600), National Natural Science Foundation of China (No. U19B2019, 61832007, 61621091), Beijing National Research Center for Information Science and Technology (BNRist), and Beijing Innovation Center for Future Chips.

REFERENCES

[1] K. Beckmann et al. 2016. Nanoscale hafnium oxide rram devices exhibit pulse dependent behavior and multi-level resistance capability. *Mrs Advances* (2016).
[2] Y. Cai et al. 2018. Long live time: improving lifetime for training-in-memory engines by structured gradient sparsification. In *DAC*. IEEE.
[3] Y. Cai et al. 2019. Low Bit-width Convolutional Neural Network on RRAM. *IEEE TCAD* (2019), 1–1.
[4] C. Y. Chen et al. 2015. RRAM Defect Modeling and Failure Analysis Based on March Test and a Novel Squeeze-Search Scheme. *IEEE TC* (2015).
[5] H. Chen et al. 2018. A >3GHz ERBW 1.1GS/S 8B Two-Sten SAR ADC with Recursive-Weight DAC. In *VLSI-Circuits, 2018*. 97–98.
[6] M. Cheng et al. 2017. TIME: A Training-in-memory Architecture for Memristor-based Deep Neural Networks. In *DAC, 2017*. ACM.
[7] P. Chi et al. 2016. PRIME: A Novel Processing-in-memory Architecture for Neural Network Computation in ReRAM-based Main Memory. In *ISCA, 2016*.
[8] K. D. Choo, J. Bell, and M. P. Flynn. 2016. 27.3 Area-efficient 1GS/s 6b SAR ADC with charge-injection-cell-based DAC. In *ISSCC, 2016*. 460–461.
[9] P. Gu et al. 2015. Technological exploration of RRAM crossbar array for matrix-vector multiplication. In *ASPDAC, 2015*. 106–111.
[10] Fatih Gül. 2019. Addressing the sneak-path problem in crossbar RRAM devices using memristor-based one Schottky diode-one resistor array. *Results in Physics* (2019).
[11] K. He et al. 2016. Deep Residual Learning for Image Recognition. In *CVPR, 2016*.
[12] Z. He et al. 2019. Noise Injection Adaption: End-to-End ReRAM Crossbar Non-ideal Effect Adaption for Neural Network Mapping. In *DAC, 2019*. 1–6.
[13] W. Huangfu et al. 2017. Computation-oriented fault-tolerance schemes for RRAM computing systems. In *ASPDAC*. IEEE.
[14] Kaggle et al. 2014. CIFAR-10 - Object Recognition in Images. website. (2014). https://www.kaggle.com/c/cifar-10.
[15] S. Karen et al. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *Computer Science* (2014).
[16] G. Krishnan et al. 2020. Interconnect-Aware Area and Energy Optimization for In-Memory Acceleration of DNNs. *IEEE Design and Test* (2020), 1–1.
[17] L. Kull et al. 2017. 28.5 A 10b 1.5GS/s pipelined-SAR ADC with background second-stage common-mode regulation and offset calibration in 14nm CMOS FinFET. In *ISSCC, 2017*. 474–475.
[18] Y. LeCun et al. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE, 1998*. 2278–2324.
[19] S. R. Lee et al. 2012. Multi-level switching of triple-layered TaOx RRAM with excellent reliability for storage class memory. In *VLSIT*.
[20] B. Li et al. 2014. ICE: Inline calibration for memristor crossbar-based computing engine. In *DATE*.
[21] D. Lin et al. 2016. Fixed point quantization of deep convolutional networks. In *ICML*. 2849–2858.
[22] M. Lin et al. 2018. DL-RSIM: A Simulation Framework to Enable Reliable ReRAM-based Accelerators for Deep Learning. In *ICCAD, 2018*. 1–8.
[23] Sumit K. Mandal et al. 2019. Analytical Performance Models for NoCs with Multiple Priority Traffic Classes. *ACM Trans. Embed. Comput. Syst.* 18, 5s (2019).
[24] B. Nasri et al. 2017. A 700 μW 1GS/s 4-bit folding-flash ADC in 65nm CMOS for wideband wireless communications. In *ISCAS, 2017*. 1–4.
[25] X. Peng et al. 2019. DNN+NeuroSim: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators with Versatile Device Technologies. In *IEDM, 2019*.
[26] A. Shafiee et al. 2016. ISAAC: A Convolutional Neural Network Accelerator with In-situ Analog Arithmetic in Crossbars. In *ISCA, 2016*.
[27] H. Sun et al. 2020. An Energy-Efficient Quantized and Regularized Training Framework For Processing-In-Memory Accelerators. In *ASPDAC, 2020*. 1–6.
[28] S. J. E. Wilton and N. P. Jouppi. 1996. CACTI: an enhanced cache access and cycle time model. *JSSC, 1996* 31, 5 (1996), 677–688.
[29] W. Wu et al. 2018. Suppress variations of analog resistive memory for neuromorphic computing by localizing Vo formation. *Journal of Applied Physics*.
[30] L. Xia et al. 2017. Fault-tolerant training with on-line fault detection for RRAM-based neural computing systems. In *DAC*.
[31] L. Xia et al. 2018. MNSIM: Simulation Platform for Memristor-Based Neuromorphic Computing System. *TCAD, 2018* (2018).
[32] S. Yu et al. 2012. A neuromorphic visual system using RRAM synaptic devices with Sub-pJ energy and tolerance to variability: Experimental characterization and large-scale modeling. In *IEDM*.
[33] Z. Zhu et al. 2019. A Configurable Multi-Precision CNN Computing Framework Based on Single Bit RRAM. In *DAC, 2019*. 1–6.
[34] Z. Zhu et al. 2020. MNSIM 2.0: A Behavior-Level Modeling Tool for Memristor-based Neuromorphic Computing Systems. In *GLSVLSI*.