Improving the Robustness and Efficiency of PIM-based Architecture by SW/HW Co-design

Xiaoxuan Yang
xy92@duke.edu
Duke University
Durham, NC, USA

Shiyu Li
shiyu.li@duke.edu
Duke University
Durham, NC, USA

Qilin Zheng
qilin.zheng@duke.edu
Duke University
Durham, NC, USA

Yiran Chen
yiran.chen@duke.edu
Duke University
Durham, NC, USA

ABSTRACT

Processing-in-memory (PIM) based architecture shows great potential to process several emerging artificial intelligence workloads, including vision and language models. Cross-layer optimizations could bridge the gap between computing density and the available resources by reducing the computation and memory cost of the model and improving the model's robustness against non-ideal hardware effects. We first introduce several hardware-aware training methods to improve the model robustness to the PIM device's non-ideal effects, including stuck-at-fault, process variation, and thermal noise. Then, we further demonstrate a software/hardware (SW/HW) co-design methodology to efficiently process the state-of-the-art attention-based model on PIM-based architecture by performing sparsity exploration for the attention-based model and circuit-architecture co-design to support the sparse processing.

CCS CONCEPTS

• **Hardware** → **Emerging architectures**; **System-level fault tolerance**; *Process, voltage and temperature variations*; • **Computing methodologies** → **Machine learning**; **Natural language processing**.

KEYWORDS

Processing-in-Memory, Hardware-Software Co-Design, Resistive Random Access Memory, Machine Learning, Transformer, Efficiency, Robustness, Variation

ACM Reference Format:
Xiaoxuan Yang, Shiyu Li, Qilin Zheng, and Yiran Chen. 2023. Improving the Robustness and Efficiency of PIM-based Architecture by SW/HW Co-design. In *28th Asia and South Pacific Design Automation Conference (ASPDAC '23), January 16–19, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3566097.3568358

ASPDAC '23, January 16–19, 2023, Tokyo, Japan

ACM ISBN 978-1-4503-9783-4/23/01.
https://doi.org/10.1145/3566097.3568358

1 INTRODUCTION

Advanced computing systems have been a critical enabler of the tremendous success of computationally intensive artificial intelligence (AI) models. However, in recent years, the advancements in transistor scaling have been offset by excessive heat, power consumption, and other issues related to physical limits. Furthermore, from a system standpoint, modern processor throughput is hampered by data transfer bandwidth. Computing efficiency, defined as performance per unit power/energy consumption, has emerged as an important metric for next-generation computing platforms.

To achieve better efficiency, one promising approach is to design domain-specialized accelerators with emerging nonvolatile memories owing to their superior characteristics. For instance, Resistive Random Access Memory (ReRAM), one type of promising emerging nonvolatile memories, features remarkable scalability and zero standby power thanks to its nonvolatility [27]. Moreover, the ReRAM with the cross-point array or crossbar structure is applicable for dense integration. This structure can fuse computation and memory within modern computing models and thus is beneficial to reduce the data-transfer overhead [35]. Based on this observation, a pioneering crossbar-based design has been proposed to realize matrix-vector multiplication (VMM) in the same physical space of storage. That is to say, processing-in-memory (PIM) architecture can bring significant efficiency improvement to computation tasks.

In an ideal case, the ReRAM device can be programmed to analog resistance to represent the values ranging from low resistance state (LRS) to high resistance state (HRS). However, the write driver and the sense amplifier can only support limited precision, resulting in the PIM system's limited bitwidth representation of involving elements [28]. Therefore, the quantization step is necessary, and the quantization robustness becomes a critical issue, especially for low-precision circumstances [7]. Moreover, the device's static and dynamic faults will degrade the inference accuracy of the neural network [11]. Therefore, PIM-based system robustness should be enhanced via software-hardware (SW/HW) co-design [36].

ReRAM-based PIM designs can naturally support the VMM computations, which are the dominant components in convolutional neural networks (CNNs). However, when the application involves more complicated function blocks, the ReRAM-based PIM designs should provide additional support for efficient data processing and computation. We find that attention-based networks bring unique

challenges for PIM designs. Therefore, we investigate potential PIM-based architectural solutions on attention-based models.

This paper analyzes cross-layer optimization for efficiency and robustness improvement and provides a case study for attention-based networks. The remaining parts are formulated as follows: Section 2 proposes the software optimization and hardware specialization for efficiency enhancement, and Section 3 focuses on the hardware-software co-design strategies regarding the robustness issue. Section 4 investigates a more complicated case, i.e., PIM-based designs for attention-based models, and shows the potential of PIM-based systems on broader models and applications. Section 5 concludes this work and discusses future topics.

2 EFFICIENCY ENHANCEMENT FOR PIM-BASED DESIGNS

Efficiency is the major concern for designing PIM-based architecture. Although PIM-based architecture can provide higher throughput and alleviate the impact of data movement, naïvely mapping workloads to PIM may lead to low utilization. If we cannot extract parallelism from the workload, only a small portion of PIM cells can participate in the computation, leading to low effective throughput. Moreover, since one of the operands needs to be stored in PIM before the computation begins, we need to guarantee the PIM is large enough to hold the data for each computation stage (i.e., work memory). If the required work memory is larger than the PIM capacity, off-chip accesses are required during computation, incurring extra power overhead and possibly stalling the computation process.

To enhance the efficiency of the PIM-based architecture, we introduce several cross-layer optimizations. From the software/algorithm perspective, we compress and reorganize the workload (e.g., the deep neural networks) to fit the working memory on-chip and extract sufficient parallelism, respectively. The software-level optimization takes the specification of the underlying hardware in mind and targets maximizing the actual end-to-end speedup rather than the theoretical benefits. From the hardware perspective, we explore the unique features of the workload and tailor the PIM-based designs for each workload. The cross-layer optimization opens a larger design space and makes it possible to seek a better solution.

2.1 Software Optimization for Efficiency

A lot of workloads to be deployed onto the PIM-based architecture contain redundant computations. For example, we can remove over 93% of the computations in a deep neural network by pruning the unimportant weights with negligible accuracy loss [32]. Pruning the weights can also reduce the memory capacity requirement. However, removing arbitrary weights will not benefit the efficiency. Since the basic operation of a PIM-based design is vector-matrix multiplication, the computation can be skipped only if all the weights of one row stored in the array are zeros. Thus, a high pruning ratio does not necessarily convert to savings of memory capacity or computation.

Structured pruning was adopted to address this challenge. Instead of removing individual weights, structured pruning tends to prune an entire structure of the model, e.g., the filter, convolutional kernel, or a row/column of weights. SSL[26] applies group LASSO regularizer, which is the $l1$-norm of the $l2$-norms of the structures

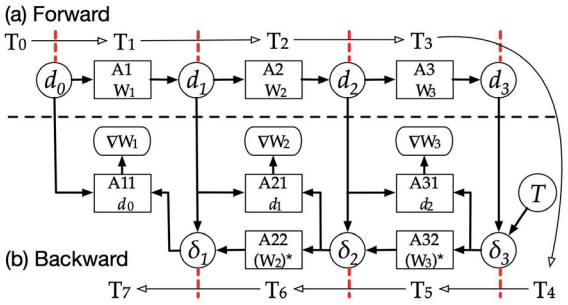


Figure 1: Pipeline configuration for DNN training in PipeLayer [22].

that we want to remove. The regularizer will drive a large number of these structures to be zeros. Apart from convolutional neural networks, this method can also be applied to other models like LSTM [39]. Structured pruning can be further enhanced for PIM-based designs. Group Scissor [24] introduces a two-step method to reduce the memory capacity and computation cost. The first step is to perform the low-rank approximation to the original weight to reduce the size of the weight, thus reducing the size of the crossbar. The second step is to perform structural pruning to remove redundant connections among crossbars and alleviate routing congestion.

Quantization is also an important method to compensate for the limited bitwidth of the memory cell. Bit-slice sparsity [40] enforces the bit-slice regularizer during quantization-aware training, producing the weights with sparse bit-slices. The all-zero bit-slice can be skipped, thus reducing the number of cells required for deployment. BSQ [31] extends this regularizer to generate bit-level sparsity, making it possible to explore a broader design space for mix-precision quantization.

2.2 Hardware Specialization for Efficiency

From the hardware perspective, we are seeing two major challenges in enhancing the efficiency, 1) finding the optimal dataflow and mapping scheme for both inference and training tasks of deep neural networks (DNNs); 2) incorporating support for emerging models/operators. We proposed a variety of PIM-based designs to address these challenges.

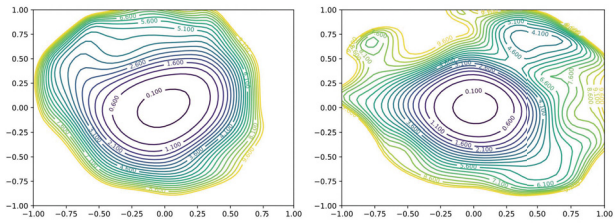
RENO [18] pioneered the adoption of ReRAM crossbars for computing neural networks. The crossbar arrays are used to calculate the vector-matrix multiplication in the analog domain. A mixed-signal router design was proposed to transfer the intermediate data in analog form while maintaining the control and routing data in digital form. RENO also features a pipeline between the PIM accelerator and CPU to execute the instructions. ISAAC [21] proposes a full-fledged design to support the inference of convolutional neural network. The read-out, sample and hold, analog-digital conversion, and post processing are scheduled in a pipeline manner in each compute tile to provide higher throughput. PRIME [3] presents a full-stack solution including hardware architecture and software interface to support *in-situ* computation of neural networks in main momory. PipeLayer [22] then introduces the support for DNN training by adding weight update into the computation pipeline and resolving the dependencies in training. The ReRAM arrays can be

configured into morphable subarrays and memory subarrays, for computation and storage, respectively. PipeLayer build a pipeline, as shown in Figure 1, to efficiently support DNN training. A portion of subarrays are configured into memory mode to store the activations for calculating the errors in the back propagation. The input samples in the same batch are processed in a pipeline fashion since the weight update is applied at the end of each batch and there is no dependency among these samples. AtomLayer [20] further addressed the inefficiency of PipeLayer for inference tasks. AtomLayer chooses to dedicate the PIM accelerator for processing one layer and uses off-chip DRAM to store the intermediate data. The extra communication with off-chip memory brings a low on-chip buffer overhead, low latency, and high hardware utilization. Lattice [41] calculates the dot product between input feature maps and the weights in the peripheral circuits to eliminate the analog-digital conversion cost.

Looking beyond conventional DNN, there are new operators and workloads that expose opportunities for PIM-based designs. Generative Adversarial Networks (GAN) involve transposed convolutional (TCONV) layers. TCONV adds a zero insertion stage before regular convolution to expand the input feature map. These inserted zeros form a subset of patterns for the convolution since only non-zero value needs to be computed. ReGAN [1] and ZARA[2] are built on top of this observation. They pre-classify the weight kernels into multiple subsets and only invoke computation between the input and relevant subsets in both the forward and backward phases of a TCONV. Depthwise separable convolution significantly reduces the parallelism by decomposing the original convolution into depthwise and pointwise stages. The lack of parallelism leads to poor efficiency on PIM. MobiLattice [42] address this issue by introducing an additional digital mode to the PIM design where the weights are read from the array and processed in a local processing element (PE). The digital mode is able to avoid excessive ADC and crossbar latency when the parallelism is insufficient to offset the cost. Apart from DNN, graph analysis is also a series of important workloads. Due to the sparse nature of the graphs, it is challenging to map such workloads to PIM-based designs. GraphR [23] implements the graph processing with each ReRAM crossbar executing the sparse matrix-vector multiplication. The graph is partitioned into subgraphs and stored in a compressed format. After that, the subgraphs are processed by the graph engines (GE) with a stream-apply execution model.

3 ROBUSTNESS IMPROVEMENT IN PIM-BASED ARCHITECTURE

To realize robustness improvement, methodologies for SW/HW co-design require minimum hardware overhead to prevent non-ideal properties from compromising the accuracy and performance of PIM systems [29]. The PIM implementation of computational networks can tolerate some level of variation, as shown in selected small networks [9]. However, the variation in the PIM-based system may have a more severe impact on large-scale networks, such as more than 10% of accuracy degradation on the VGG19 network on the CIFAR-10 dataset [12]. For the PIM system variation, we consider the cell-level, subarray-level, and system-level variation [35]. For example, the stuck-at fault [8] is viewed as cell-level variation;



(a) HERO, test accuracy: 93.44% (b) SGD, test accuracy: 92.82%

Figure 2: Loss surface contour along two random directions around converged weights. Estimated on ResNet20 model on CIFAR-10 dataset trained with HERO [33] and SGD.

IR-drop [10, 15] is categorized as subarray-level variation; process variation is considered as system-level variation. Furthermore, we find that several strategies are effective in robustness improvement to these variations, such as the retraining method during finetune stage [17], error compensation with redundant crossbar/array [15], and inline calibration scheme [19]. To enhance the robustness of PIM systems, we first look into the hardware design and algorithm optimization for quantization robustness and provide promising solutions. Moreover, for robustness against the variation, we investigate the hardware-aware training method for inference and training stages and discuss reliable and effective hardware design.

3.1 Design and Optimization for Quantization Robustness

As mentioned in Section 2.1, quantization is an effective method to save memory bandwidth. However, due to the fixed precision quantization, the weight will be perturbed by a l_∞ bounded variation, deviating the result from the software training result. Moreover, limited memristor resolution leads to reduced precision of synaptic weights and drastically reduces system accuracy. To address this issue, a hardware/software co-design process that adapts training iterations to generate hardware-compatible weights directly is highly effective. The following are three techniques in the co-design process [25]: Firstly, distribution-aware quantization selects the optimum discrete weight values based on the device-programmed memristance distribution. Additionally, quantization regularization ensures that well-trained weights are distributed in the manner provided by the quantization scheme. Furthermore, independent bias tuning assists neural networks in learning the optimal bias to compensate for the accuracy offsets caused by distribution-aware quantization and quantization regularization. It also helps to reduce the impact of process variation.

Furthermore, a recent study uncovers that quantization robustness and generalization performance can be unified and optimized under the Hessian-enhance regularization optimization (HERO) [33]. HERO follows the observation that the generalization gap between training and inference is bounded by a l_2 weight perturbation [5]. In order to jointly minimize the generalization gap and quantization loss, the Hessian value is proved to be necessary for the gradient formulation. As a result, HERO can achieve a more flat loss surface around the converged weights as shown in Figure 2. HERO can boost the quantization robustness and will benefit the PIM deployment accuracy under various weight precision settings.

Table 1: In-memory training accuracy results under ReRAM programming variation in ResNet20 network on CIFAR-10 dataset with SGD and ESSENCE [38].

Method	0.5%	1.0%	1.5%	2.0%
SGD	82.79%	37.40%	23.95%	17.03%
ESSENCE	89.71%	88.21%	83.07%	76.73%

3.2 Hardware-aware Training for Variation Robustness

Aside from the quantization variation, the PIM system also suffers from process variation, input signal noise, thermal noise, shot noise, and other types of noise [4]. To combat the process variation and input signal noise, the noise-eliminating training (NET) method [14] based on a new crossbar structure is proposed to eliminate the noise accumulation during training. NET method can adapt dynamic threshold to reduce the noise amplitude in the high-sensitivity region of two selected activation functions, namely *sigmoid* and *sgn* function. In general, the optimal selection of dynamic threshold can help reduce the PIM system sensitivity to noise without the failure of training convergence. Moreover, the variation-aware training method [16] can actively compensate the impact of device variations and optimizes the mapping scheme from computations to crossbars.

However, due to the computing efficiency consideration, the high operating frequency of PIM system is preferable. At the same time, the amplitude of the thermal noise and shot noise would increase, and lead to the convergence failure of the previous training method [6]. In order to combat the combination of thermal noise and shot noise at the high operating frequency, we propose the ReRAM-based stochastic-noise-aware training (ReSNA) method [34]. The ReSNA method starts with the analysis of the noise distribution at specific operating frequency and temperature. The noise amplitude is then converted to the relative impact on the weight value. After that, such relative impact and other PIM hardware settings are included in the training process. As a result, ReSNA can provide reliable and robust training results under the combination of noise. More importantly, ReSNA is a generalized method as it can also provide promising inferencing results even when further including random telegraph noise and programming noise.

The NET and ReSNA methods consider the PIM inference stage and provide effective hardware-aware training solutions. But note that PIM-based training is a more challenging problem under the consideration of variation. The main reason is that each training iteration involves the reprogramming of the weight value, and thus introduces programming variation periodically. Even with batch training techniques, PIM-based training is still unstable and may fail. Recently, an endurance-aware training method ESSENCE [38] proposes to utilize a structured sparse gradient matrix to update the weight and reduce the number of reprogramming by up to 10×. One impressive benefit of ESSENCE is more stable training results under the programming variation. Under 2.0% Gaussian distributed random programming noise, ESSENCE can outperform SGD by more than 59%, as indicated in Table 1. Therefore, ESSENCE enables robust and reliable in-memory training against the variation.

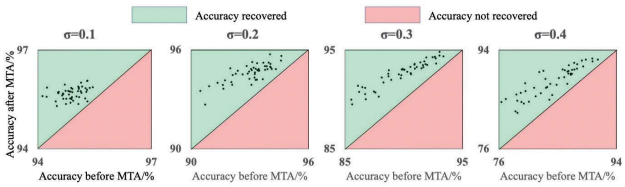


Figure 3: Accuracy recovered by MFTA [13] under various process variations (denoted by σ). x- and y -axis are the accuracy before and after applying MFTA, respectively.

3.3 Hardware Exploration for Variation Robustness

For the hardware design, we aim to stabilize the weight representation and improve the PIM-system robustness. To provide a comprehensive analysis, we cover two circumstances for the PIM implementations of multi-layer perceptron (MLP) and spiking neural network (SNN), where the input value is encoded as the voltage amplitude and spike firing time, respectively.

For the MLP case, the integration of the feedback controller in [30] to adaptively modify the sensor directions significantly reduces this accuracy loss. The feedback controller chooses the next inferencing voltage direction depending on the results of the previous round of inferencing rather than using the "compute-and-read-verify" method. By doing this, the PIM engine develops into a self-correcting system that can handle any faults that may be brought on by the read disturbance. When compared to the naïve open loop design without any feedback controllers, the service life (which represents the length of reliable recall executions) with a feedback controller is increased by a factor of over ten.

For the SNN case, the robustness improvement involves the adjustment of the firing threshold and timing threshold. For instance, variation has an impact on the post-spike firing rates from the current-based activation function, and subsequently reduces computational accuracy. By tuning firing thresholds to alter the firing times, this issue can be successfully addressed. The multi-level firing threshold adjustment (MFTA) method intentionally increases the firing thresholds of post-neuron post-spikes when they fire earlier than anticipated due to ReRAM process variations and vice versa [13]. The results in Figure 3 are generated from the Monte Carlo simulation to validate the efficacy of the MFTA scheme. The MTFA can recover the accuracy under various variation levels and maintain high inferencing accuracy results. In summary, hardware designs for different networks can provide promising solutions for better variation robustness.

4 CASE STUDY FOR ATTENTION-BASED MODEL

4.1 Dense Attention-based Model

Attention-based neural networks have shown superior performance in a wide range of tasks. However, the computational complexity of the attention operation hinders the deployment of attention-based neural networks on resource-constrained devices. Although researchers have successfully applied ReRAM-based PIM to accelerate conventional neural networks, the unique computation process

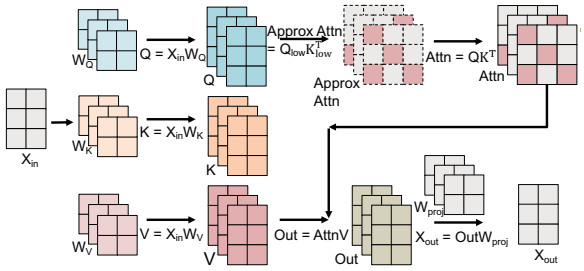


Figure 4: Illustration of sparse attention.

of the scaled dot-product attention makes it difficult to be directly used in these designs. The key challenge for scaled dot-product for ReRAM-based implementation is the intermediate result challenge. In the attention-based model, the two operands Q and K^T are both obtained as intermediate results generated from previous linear layers. To calculate the subsequent Q and K^T, one of these two matrices has to be loaded to the ReRAM array as a conductance matrix to perform PIM MatMul. However, constrained by the programming driver circuits, programming a matrix into a ReRAM array is usually slow, and it could cause a longer latency to load intermediate results into ReRAM arrays and stall the computation. Moreover, the iterative calculation in a multi-head self-attention module inevitably causes frequent rewriting of ReRAM cells, leading to a high and unavoidable write energy consumption.

ReTransformer [37] proposes a pipelined method to reorder data generation and computation process of scaled dot-product attention layer between W_Q and W_K. At beginning of the inference, W_Q and W_K are initialized and written into two ReRAM crossbars, respectively. To perform the dot product between Q and K in the ReRAM-based PIM module, one of the intermediate results Q and K, must be stored in a ReRAM crossbar.

The basic method is to decompose the computation into two cascaded multiplication steps. The original computation order is to compute two weights matrices with input matrices to generate Q and K, and then compute the Q and K^T multiplication. An optimized computation flow is to compute the Q matrix first. Then, instead of computing K matrix, the computation is reordered to perform the multiplication between Q and weight. The generated output is then multiplied by input matrix. Hence, a possible way to avoid writing the intermediate results into a ReRAM crossbar can be the follows: at the beginning of the process, we initialize ReRAM crossbars with the weight matrices. Then we could form a pipeline where the first stage is to compute Q by multiplying weights and input, and the second stage is to compute intermediate results by multiplying Q and weight. In the meanwhile, we initialize another ReRAM crossbar which stored the input matrix. We then perform could compute the intermediate result with the input matrix to save processing time. The evaluation result shows that the proposed pipeline scheme could improve efficiency by up to 80%.

4.2 Sparse Attention-based Model

From the algorithm perspective, a better accuracy-computation complexity trade-off can be achieved by exploiting a dynamic and unstructured sparsity in the attention map matrix (as shown in Figure 4), where the original attention computation is converted to a sampled dense-dense matrix multiplication (SDDMM) and a sparse-dense matrix multiplication (SpMM). However, the unique unstructured and dynamic sparsity pattern in the sparse attention challenges the mapping efficiency of the PIM architecture, as the conventional PIM architecture uses a VMM primitive. For one PIM bank, the sparse computation pattern leads to a low utilization rate, which originates from rigid input/output dimensions of the VMM primitive. We can abstract each PIM bank into a $M \times N$ PE array which computes the inner product between a $M \times N$ matrix and a $1 \times M$ vector in each cycle. in the attention map computation stage, the k vectors are stored in the memory array, and the Q vectors are streamed into the PIM bank, where the output attention map matrix is streamed out. At each cycle, M elements in a Q vector can be computed with N K vectors. In this case, the PIM bank is fully utilized only when N continuous indexes in the attention map mask are valid. In other words, the output dimension of the PIM bank cannot be fully filled, which will reduce the effective output parallelism. Similarly, for the output computation stage, the V vectors are stored in the memory array and the sparse attention map matrix is streamed into the PIM bank. At each cycle, M elements in a row of the attention map are computed with N V vectors. We need to ensure N continuous indexes in the attention map mask are valid to fully utilize the PE array. That is to say, the input dimension is not fully filled with a reduction of the effective input parallelism. In general, for the current PIM architecture, the computation primitive provided by the PIM bank is not suitable for sparse processing in the attention model.

We would like to address this problem by performing a circuit-architecture co-design methodology. The conventional PIM bank only provides the primitive of VMM with fixed input and output parallelism. To improve the mapping efficiency, the basic idea is to collapse the input or output dimensions according to the presence of sparsity. We still abstract each PIM bank into a $M \times N$ PE array. However, we assume that the PE array can be dynamically configured to compute the inner product between two $1 \times MN$ vectors or multiplication between one scalar and one $1 \times MN$ vectors. In the attention map computation stage, the output dimension is sparse. Instead of mapping the sparse dimension into the spatial parallelism, we map the sparse dimension in a temporal loop. For each computation cycle, we compute one non-zero value in the sparse attention map from the inner product between one q and k vectors. The computation corresponding to the pruned attention scores can be skipped. Since the token dimension is a relatively large value (e.g. 64, 128) and M, N is a small value (e.g. 8), the $M \times N$ PE array can be fully utilized. Similarly, when the input dimension is sparse, the input is viewed as a scalar and each computation cycle produces one output vector corresponding to the product of the non-zero attention map and one V vector. For each computation cycle, we feed one non-zero value in the sparse attention map as a scalar and perform a scalar vector multiplication between the scalar and one specific v vector. The computation corresponding to the zero-value attention map can be skipped.

Following this micro-arch design principle, we need to have some requirements of the PIM bank design methodology, where each bank should be reconfigured to support one of these three primitives. In scalar vector multiplication (SVM) mode, the PIM bank receives

a scalar as the input, multiplies it by the vector stored in the bank, and produces a vector as the output. In the inner product (IP) mode, the PIM bank receives a vector as the input, computes the dot-product of the input vector and the stored vector, and produces a scalar as the output. In addition, the throughput delivered by each bank should be maintained as it is in the original design. Thus, we need to scale up the vector dimension in these two primitives. For example, if the original PIM bank provides the VMM primitive of the shape $M \times N$ (i.e., input length is M, output length is N), our reconfigurable PIM bank design will provide the SVM primitive with the shape of $1 \times MN$, or the IP primitive with the shape of $MN \times 1$. Therefore, the proposed design can efficiently support the unstructured and dynamic sparsity pattern in the attention map.

5 CONCLUSION

In this paper, we demonstrate that cross-layer optimizations could bridge the gap between computing density and the available resources by reducing the computation and memory cost of the model. Moreover, We introduce design and optimization to improve the model robustness to the PIM device's non-ideal effects. Furthermore, we propose a SW/HW co-design methodology to process the state-of-the-art attention-based model on PIM-based architecture efficiently. We believe that the robustness enhancement remains an open question since the theoretical analysis is worth exploring. Besides, the automatic design flow for PIM systems should also be included in the SW/HW co-design framework.

ACKNOWLEDGMENTS

This work is supported by ARO W911NF-19-2-0107, NSF 2112562, and NSF 1955246.

REFERENCES

[1] Fan Chen, Linghao Song, and Yiran Chen. 2018. ReGAN: A Pipelined ReRAM-Based Accelerator for Generative Adversarial Networks. In *ASP-DAC*. 178–183.
[2] Fan Chen, Linghao Song, Hai Helen Li, and Yiran Chen. 2019. Zara: A Novel Zero-Free Dataflow Accelerator for Generative Adversarial Networks in 3D ReRAM. In *DAC*. 1–6.
[3] Ping Chi et al. 2016. PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory. In *ISCA*. 27–39.
[4] Ben Feinberg, Shibo Wang, and Engin Ipek. 2018. Making Memristive Neural Network Accelerators Reliable. In *HPCA*. IEEE, 52–65.
[5] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2020. Sharpness-Aware Minimization for Efficiently Improving Generalization. *arXiv preprint arXiv:2010.01412* (2020).
[6] Zhezhi He, Jie Lin, Rickard Ewetz, Jiann-Shiun Yuan, and Deliang Fan. 2019. Noise Injection Adaption: End-to-End ReRAM Crossbar Non-ideal Effect Adaption for Neural Network Mapping. In *DAC*. 1–6.
[7] Jörg Henkel et al. 2022. Approximate Computing and the Efficient Machine Learning Expedition. In *ICCAD*. 9 pages.
[8] Miao Hu, Hai Li, Yiran Chen, Qing Wu, and Garrett S Rose. 2013. BSB Training Scheme Implementation on Memristor-Based Circuit. In *CISDA*. IEEE, 80–87.
[9] Miao Hu, Hai Li, Qing Wu, Garrett S Rose, and Yiran Chen. 2012. Memristor Crossbar Based Hardware Realization of BSB Recall Function. In *IJCNN*. IEEE, 1–7.
[10] Boxun Li, Yu Wang, Yiran Chen, Hai Helen Li, and Huazhong Yang. 2014. ICE: Inline Calibration for Memristor Crossbar-based Computing Engine. In *DATE*. IEEE, 1–4.
[11] Bing Li, Bonan Yan, Chenchen Liu, and Hai Li. 2019. Build Reliable and Efficient Neuromorphic Design with Memristor Technology. In *ASP-DAC*. 224–229.
[12] Ziru Li, Bonan Yan, and Hai Li. 2020. ReSiPE: ReRAM-based Single-Spiking Processing-In-Memory Engine. In *DAC*. IEEE, 1–6.
[13] Ziru Li, Qilin Zheng, Bonan Yan, Ru Huang, Bing Li, and Yiran Chen. 2022. ASTERS: Adaptable Threshold Spike-Timing Neuromorphic Design with Twin-Column ReRAM Synapses. In *DAC*. 1099–1104.
[14] Beiye Liu et al. 2013. Digital-Assisted Noise-Eliminating Training for Memristor Crossbar-based Analog Neuromorphic Computing Engine. In *DAC*. IEEE, 1–6.
[15] Beiye Liu et al. 2014. Reduction and IR-drop Compensations Techniques for Reliable Neuromorphic Computing Systems. In *ICCAD*. IEEE, 63–70.
[16] Beiye Liu, Hai Li, Yiran Chen, Xin Li, Qing Wu, and Tingwen Huang. 2015. Vortex: Variation-aware Training for Memristor X-bar. In *DAC*. 1–6.
[17] Chenchen Liu, Miao Hu, John Paul Strachan, and Hai Li. 2017. Rescuing Memristor-based Neuromorphic Design with High Defects. In *DAC*. IEEE, 1–6.
[18] Xiaoxiao Liu et al. 2015. RENO: A High-Efficient Reconfigurable Neuromorphic Computing Accelerator Design. In *DAC*. Association for Computing Machinery, Article 66, 6 pages.
[19] Xiaoxiao Liu et al. 2016. Harmonica: A Framework of Heterogeneous Computing Systems with Memristor-based Neuromorphic Computing Accelerators. *TCAS-I* 63, 5 (2016), 617–628.
[20] Ximing Qiao, Xiong Cao, Huanrui Yang, Linghao Song, and Hai Li. 2018. Atomlayer: A Universal ReRAM-Based CNN Accelerator with Atomic Layer Computation. In *DAC*. Association for Computing Machinery, Article 103, 6 pages.
[21] Ali Shafiee et al. 2016. ISAAC: A Convolutional Neural Network Accelerator with in-Situ Analog Arithmetic in Crossbars. In *ISCA*. IEEE Press, 14–26.
[22] Linghao Song, Xuehai Qian, Hai Li, and Yiran Chen. 2017. PipeLayer: A Pipelined ReRAM-Based Accelerator for Deep Learning. In *HPCA*. 541–552.
[23] Linghao Song, Youwei Zhuo, Xuehai Qian, Hai Li, and Yiran Chen. 2018. GraphR: Accelerating Graph Processing using ReRAM. In *HPCA*. IEEE, 531–543.
[24] Yandan Wang, Wei Wen, Beiye Liu, Donald Chiarulli, and Hai Li. 2017. Group Scissor: Scaling Neuromorphic Computing Design to Large Neural Networks. In *DAC*. IEEE, 1–6.
[25] Yandan Wang, Wei Wen, Linghao Song, and Hai Helen Li. 2017. Classification Accuracy Improvement for Neuromorphic Computing Systems with One-Level Precision Synapses. In *ASP-DAC*. IEEE, 776–781.
[26] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning Structured Sparsity in Deep Neural Networks. *NeurIPS* 29 (2016).
[27] Bonan Yan et al. 2019. Resistive Memory-Based In-Memory Computing: From Device and Large-Scale Integration System Perspectives. *Adv. Intell. Syst.* 1, 7 (2019), 1900068.
[28] Bonan Yan et al. 2019. RRAM-based Spiking Nonvolatile Computing-In-Memory Processing Engine with Precision-Configurable In Situ Nonlinear Activation. In *Symp. VLSI Technol.* IEEE, T86–T87.
[29] Bonan Yan, Ziru Li, Brady Taylor, Hai Li, and Yiran Chen. 2020. Neuromorphic Computing Systems with Emerging Nonvolatile Memories: A Circuits and Systems Perspective. In *VLSI-TSA*. IEEE, 122–123.
[30] Bonan Yan, Jianhua Yang, Qing Wu, Yiran Chen, and Hai Li. 2017. A Closed-Loop Design to Enhance Weight Stability of Memristor Based Neural Network Chips. In *ICCAD*. IEEE, 541–548.
[31] Huanrui Yang, Lin Duan, Yiran Chen, and Hai Li. 2021. BSQ: Exploring Bit-Level Sparsity for Mixed-Precision Neural Network Quantization. In *ICLR*.
[32] Huanrui Yang, Wei Wen, and Hai Li. 2019. DeepHoyer: Learning Sparser Neural Network with Differentiable Scale-Invariant Sparsity Measures. In *ICLR*.
[33] Huanrui Yang, Xiaoxuan Yang, Neil Zhenqiang Gong, and Yiran Chen. 2022. HERO: Hessian-Enhanced Robust Optimization for Unifying and Improving Generalization and Quantization Performance. In *DAC*. 25–30.
[34] Xiaoxuan Yang et al. 2021. Multi-objective Optimization of ReRAM Crossbars for Robust DNN Inferencing under Stochastic Noise. In *ICCAD*. IEEE, 1–9.
[35] Xiaoxuan Yang, Brady Taylor, Ailong Wu, Yiran Chen, and Leon O Chua. 2022. Research Progress on Memristor: From Synapses to Computing Systems. *TCAS-I* 69, 5 (2022), 1845–1857.
[36] Xiaoxuan Yang, Changming Wu, Mo Li, and Yiran Chen. 2022. Tolerating Noise Effects in Processing-in-Memory Systems for Neural Networks: A Hardware–Software Codesign Perspective. *Adv. Intell. Syst.* (2022), 2200029.
[37] Xiaoxuan Yang, Bonan Yan, Hai Li, and Yiran Chen. 2020. ReTransformer: ReRAM-based Processing-in-Memory Architecture for Transformer Acceleration. In *ICCAD*. 1–9.
[38] Xiaoxuan Yang, Huanrui Yang, Janardhan Rao Doppa, Partha Pratim Pande, Krishnendu Chakrabarty, and Hai Li. 2022. ESSENCE: Exploiting Structured Stochastic Gradient Pruning for Endurance-aware ReRAM-based In-Memory Training Systems. *TCAD* (2022), 13 pages.
[39] Jingchi Zhang, Jonathan Huang, Michael Deisher, Hai Li, and Yiran Chen. 2020. Structural Sparsification for Far-Field Speaker Recognition with Intel® Gna. In *ICASSP*. IEEE, 3037–3041.
[40] Jingyang Zhang, Huanrui Yang, Fan Chen, Yitu Wang, and Hai Li. 2019. Exploring Bit-Slice Sparsity in Deep Neural Networks for Efficient ReRAM-Based Deployment. In *EMC2-NIPS*. 1–5.
[41] Qilin Zheng et al. 2020. Lattice: An ADC/DAC-less ReRAM-based Processing-In-Memory Architecture for Accelerating Deep Convolution Neural Networks. In *DAC*. 1–6.
[42] Qilin Zheng et al. 2020. MobiLattice: A Depth-wise DCNN Accelerator with Hybrid Digital/Analog Nonvolatile Processing-In-Memory Block. In *ICCAD*. 1–9.