HyperX: A Hybrid RRAM-SRAM partitioned system for error recovery in memristive Xbars

Adarsh Kosta*, Efstathia Soufleri*, Indranil Chakraborty, Amogh Agrawal, Aayush Ankit, Kaushik Roy
Purdue University, West Lafayette, USA {akosta, esoufler, ichakra, agrawa64, aankit, kaushik}@purdue.edu
**Authors contributed equally*

***Abstract*—Memristive crossbars based on Non-volatile Memory (NVM) technologies such as RRAM, have recently shown great promise for accelerating Deep Neural Networks (DNNs). They achieve this by performing efficient Matrix-Vector-Multiplications (MVMs) while offering dense on-chip storage and minimal off-chip data movement. However, their analog nature of computing introduces functional errors due to non-ideal RRAM devices, significantly degrading the application accuracy. Further, RRAMs suffer from low endurance and high write costs, hindering on-chip trainability. To alleviate these limitations, we propose HyperX, a hybrid RRAM-SRAM system that leverages the complementary benefits of NVM and CMOS technologies. Our proposed system consists of a fixed RRAM block offering area and energy-efficient MVMs and an SRAM block enabling on-chip training to recover the accuracy drop due to the RRAM non-idealities. The improvements are reported in terms of energy and product of latency and area ($ms \times mm^2$), termed as area-normalized latency. Our experiments on CIFAR datasets using ResNet-20 show up to 2.88× and 10.1× improvements in inference energy and area-normalized latency, respectively. In addition, for a transfer learning task from ImageNet to CIFAR datasets using ResNet-18, we observe up to 1.58× and 4.48× improvements in energy and area-normalized latency, respectively. These improvements are with respect to an all-SRAM baseline.**

***Index Terms*—Hybrid, RRAM, SRAM, NVM, crossbar, energy-efficiency, transfer-learning**

I. Introduction

Deep Neural Networks (DNNs) have recently shown immense promise in various applications such as computer vision and natural language processing (NLP). However, to attain progressively better algorithmic performance, DNN models have rapidly grown in size and complexity, placing enormous strain on current hardware technologies. This has led to a growing interest in domain-specific accelerators for DNN workloads. To address this, researchers have developed domain-specific accelerators such as Google's TPU [1] and Microsoft's Brainwave [2] which aim to offer efficient matrix-vector-multiplication (MVM) operations - the key computational kernel in DNNs. Since the storage and computation requirements for DNNs are growing at a faster rate than improvements in DNN accelerators [3], there is a growing interest in alternate computing architectures using NVM technologies such as RRAM [4].

RRAM offers high on-chip density owing to the smaller bit-cell area compared to SRAM and the possibility of storing multi-level states. RRAM devices can be arranged in a crossbar-like fashion to perform fast and efficient MVMs inside the memory array itself [5]. The increase in storage density significantly reduces off-chip DRAM accesses, which in turn lead to multi-fold improvements in energy and latency over their digital counterparts [6]. This has heavily encouraged research towards the development of NVM based accelerators such as ISSAC [7] and PUMA [5].

Despite their promises, the analog nature of computation in RRAM crossbars leads to functional errors in the MVM arithmetic. These errors are caused due to non-idealities associated with the RRAM devices such as parasitic resistances, non-linear device characteristics, sneak paths, etc. The resultant deviations in the MVM outputs lead to severe degradation in application accuracy [8]. To prevent this degradation, re-training of the weights mapped on crossbars is necessary [9], [10]. However, the associated high write cost and poor write endurance [11] hinder the ability of on-chip weight updates within such systems.

To that effect, we propose HyperX, a hybrid RRAM-SRAM system that leverages the benefits offered by both RRAM and SRAM memories in a complementary fashion. The RRAM block of our hybrid system leads to fast and efficient MVMs along with offering high on-chip density. In contrast, the SRAM block enables on-chip training to recover the accuracy drop caused by RRAM non-idealities.

The main contributions of this work are as follows:

- We propose a hybrid RRAM-SRAM system where a DNN is layer-wise partitioned into an RRAM block with fixed weights and an SRAM block with trainable weights.
- We provide analyses for different partitioning configurations of RRAM and SRAM blocks considering various crossbar sizes and weight/activation precision.
- We perform experiments for an inference task using ResNet-20 on CIFAR datasets and a transfer learning task using ResNet-18 from ImageNet to CIFAR datasets. Our results show that a mindful partitioning of a DNN into RRAM and SRAM blocks can lead to promising energy and area-normalized latency improvements while maintaining accuracy comparable to an all-SRAM baseline.

II. Related Work

Past works have explored modeling the effect of crossbar non-idealities and subsequently mitigating them. These include variation and technology aware training [9], [12], and local learning [13]. In addition, some works explore post-processing-based compensation by modifying RRAM weights [14]. In contrast, alternative approaches explore hybrid systems of analog and digital accelerators. These range from using NVM devices

978-3-9819263-6-1/DATE22/©2022 EDAA

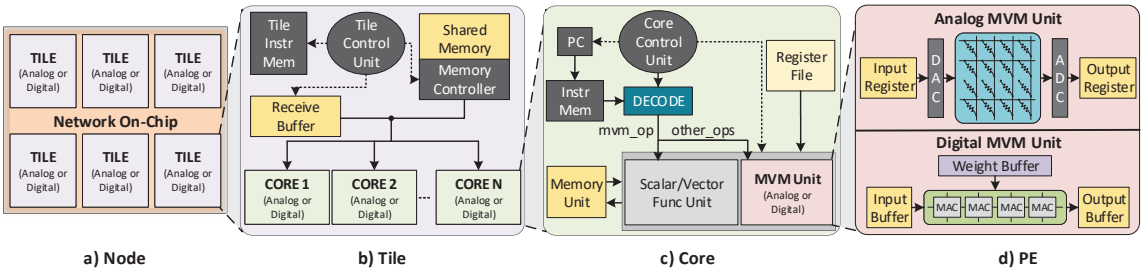

Fig. 1. a) Node architecture. b) Tile architecture. c) Core architecture. d) Analog and digital PEs. These architectures are based on PUMA [5].

as storage cells for fixed DNN weights [15], to techniques targeting recovery of certain errors associated with NVM compute [16]–[18].

Our proposed HyperX system provides a generic architecture for flexible layer-wise partitioning of any DNN model on analog and digital hardware. It eliminates the need for any architectural modifications on the original DNN model, such as additional trainable parameters or adjustable offsets used in the above works. The partitioning configuration enables attaining desired efficiency-accuracy tradeoffs while assuring high generalization capability to all tasks. The effects of non-idealities are captured using the crossbar model from [8]. The proposed approach not only offers accuracy recovery in the presence of non-idealities for standard DNN inference but also enables NVM compute for applications such as transfer learning which necessarily demands trainability.

III. Hybrid RRAM-SRAM System

A. HyperX System Architecture

Our proposed hybrid RRAM-SRAM system has both analog computing cores based on RRAM crossbars and digital computing cores based on fixed-point multiply-and-accumulate (MAC) units. The microarchitectures of both analog and digital cores are based on the PUMA hardware architecture [5]. Fig. 1 shows the complete architecture (node, tile, core, and processing elements (PE)) of the hybrid hardware. The analog and digital cores have various common components to manage instruction set execution, memory access, and other functional units to carry out scalar/vector operations. The main processing elements (PEs) in the two types of cores are described next.

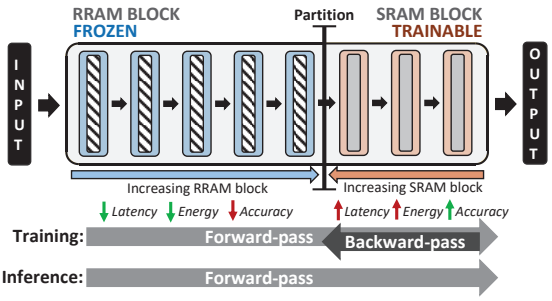

Fig. 2. A DNN partitioned into an RRAM block and an SRAM block.

1) Analog MVMU: The Analog MVMU performs MVM operation using RRAM crossbars. It interfaces with the external memories in the core using input and output registers. The DACs and ADCs handle the data conversion at the input and output registers, respectively. Each MVMU represents a logical crossbar containing multiple physical crossbars. For an RRAM device with P-bits of multi-level cell (MLC) storage and weights having K-bit precision, an MVMU contains K/P number of crossbars. The K-bit inputs are streamed in through the DACs in a bit-serial manner, 1-bit at a time. Each MVMU performs an MVM operation between 1×N K-bit input vector and N×N K-bit weight matrix. The entire computation at a logical crossbar is realized by each physical crossbar performing smaller-bit operations and generating partial outputs, which are accumulated across bits using shift-and-add registers.

2) Digital MVMU: The Digital MVMU performs MVM operation using a systolic array of Multiply-and-Accumulate (MAC) units. It consists of an input buffer storing a 1×N K-bit input vector and a weight buffer storing an N×N K-bit weight matrix. Each digital MVMU consists of N K-bit fixed-point MAC units. An N×1 sized row vector is read sequentially from the weight buffer and multiplied with the input vector using the MAC units to generate an element of the output. This is done for all rows of the weight matrix for generating the 1×N output vector which gets stored in the output buffer.

A core, whether analog or digital, can have multiple respective MVMUs to enable further parallelism for large MVM operations. Multiple cores of the same type (analog or digital) constitute a tile and are connected to a shared memory. The analog and digital tiles are connected with a network-on-chip to constitute a node. The proposed hybrid RRAM-SRAM system is considered as a fully-spatial node architecture allowing DNN models to be mapped entirely onto the on-chip hardware without requiring any off-chip accesses. This leads to low-latency and high-area design. Such a system can also be designed using an off-chip DRAM and smaller on-chip memory with higher latency and lower area. As a result, evaluating the product of latency and area ($ms \times mm^2$), termed as area-normalized latency (ANL) is more sensible for such hybrid systems.

A forward pass would require partitioning the specification of the DNN workload, i.e., the instructions for which layers are to be executed using the analog tiles and which are to be executed using the digital tiles. Each layer may require multiple tiles to map, depending on the size of the layer.

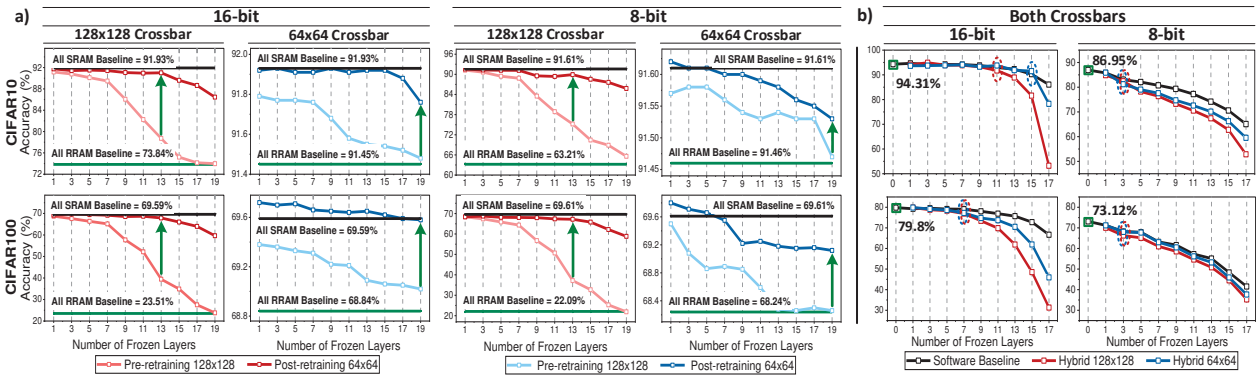

Fig. 3. a) Inference using ResNet-20 and b) Transfer learning using ResNet-18. The green arrows indicate the highest freeze configuration for full accuracy recovery. The red and blue ovals indicate the highest freeze configuration with acceptable accuracy for the two crossbar sizes. Best viewed in color.

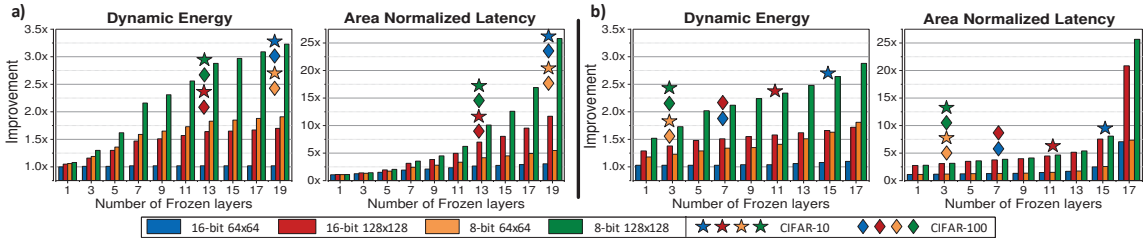

Fig. 4. Energy and Area-Normalized Latency (ANL) improvements for the Hybrid system. a) Inference with ResNet-20 and b) Transfer learning with ResNet-18. The star (diamond) label denote the number of frozen layers for best tradeoffs on CIFAR-10(100). Best viewed in color.

B. DNN workload mapping

Once a DNN is trained, it is partitioned into two contiguous RRAM and SRAM blocks (Fig. 2). The RRAM block is hosted on the analog tiles and consists of initial layers of the DNN. The SRAM block consists of the remaining layers of the DNN, hosted on the digital tiles. During the forward pass, the input is fed through the RRAM block and the computations occur in the analog MVMUs. The output activations from this block are then passed on to the SRAM block, which employs digital MVMUs to compute the final output. During re-training, the error gradients are backpropagated only for layers belonging to the SRAM block. The corresponding weights on the digital tiles are then subsequently updated.

For an inference application, the erroneous computations due to device and circuit non-idealities in the RRAM block can severely impact the application accuracy [8]. However, since the weights in the layers corresponding to the SRAM block can be updated, having a one-time re-training phase after the model deployment can help recover the accuracy degradation. In addition, our proposed hardware system allows on-chip training applications, such as transfer learning [19]. In transfer learning, parameters from a DNN model trained on a generic source task are used to learn a specific target task, as long as the two tasks are similar in nature. The initial layers of the DNN model can act as generic feature extractors and constitute the RRAM block. The remaining layers comprise the SRAM block and not only recover the errors due to analog computing but also get trained on the target task.

The energy and area benefits that can be attained through partition mapping largely depend on its layer characterization determined by the kernel size, input and output channels, activation size, etc. Each layer has a memory space and computational requirement determined by the number of parameters and number of operations during the forward pass, respectively. Partitioned mapping leads to a freeze configuration denoted by *freeze-**x***. For such a mapping, the first ***x*** layers are fixed on RRAM while the rest of the layers are on SRAM.

IV. Experimental Results

A. Experimental Setup

HyperX is evaluated using the PUMA performance [5] and functional simulator in conjunction with GENIEx [8], which considers non-idealities in crossbar evaluation. We use K=16,8 bits for input and weight precision and P=2 bits of MLC storage in the RRAM device. We perform experiments with 128×128 and 64×64 crossbars. All inference operations are fixed-point and all pre-training, and re-training experiments are floating-point. For 8-bit precision, quantization aware re-training is carried out. The ADC used in the analog MVMUs has a 8(7)-bit resolution for 128×128(64×64) crossbar size.

We target image classification tasks using ResNets [20] on CIFAR and ImageNet datasets. As mentioned in Section III-B, we partition the pre-trained DNN models into analog (RRAM) and digital (SRAM) tiles of the hardware. During re-training of the SRAM block, a learning rate in the range $[0.1 - 0.001]$ in conjunction with a multi-step learning rate schedule was used. This is done to achieve algorithmic convergence. The SRAM block was trained for 50 epochs with a batch-size of 128,

cross-entropy loss, and a Stochastic Gradient Descent (SGD) optimizer. For dynamic energy and ANL comparisons, we consider a fully-spatial all-SRAM baseline based on PUMA [5].

B. Simulation Results

1) Inference: Fig. 3 (a) illustrates the accuracy, both after mapping (pre-retraining) and subsequent retraining (post-retraining) of the SRAM block using ResNet-20.

Pre-retraining (*Light colored plots*) Going from left to right, the accuracy degradation increases for all cases. For different crossbar sizes with the same bit-precision (*red vs blue plots*), the impact is more severe for crossbar size of 128×128, due to higher circuit non-idealities.

Post-retraining (*Dark colored plots*) For 64×64 crossbar, re-training recovers the accuracy drop entirely for all freeze configurations and both bit-precisions. For 128×128 crossbar, accuracy is recovered up to *freeze-**13*** for both bit-precisions.

Fig. 4(a) plots the improvements in dynamic energy and ANL for different crossbar sizes and bit-precisions using ResNet-20. With 128×128 crossbar, the *freeze-**13*** configuration achieves approximately 1.64× (2.88×) and 7.0× (10.1×) improvement in energy and ANL, respectively, for 16 (8)-bit precision. With 64×64 crossbars, since the accuracy is recovered for all freeze configurations, *freeze-**19*** provides the optimal trade-off with 1.02× (1.91×) and 3.06× (5.47×) energy and ANL improvements, respectively, for 16 (8)-bit precision.

2) Transfer Learning: Fig. 3 (b) shows the accuracy for ResNet-18 transfer learned from Imagenet to CIFAR-10(100).

For 16-bit precision on CIFAR-10, the hybrid systems with 128×128 and 64×64 crossbars are able to achieve software accuracy up to *freeze-**11*** and *freeze-**15*** configurations, respectively. While for CIFAR-100, both crossbar sizes show accuracy degradation after *freeze-**7*** configuration.

For 8-bit precision, there is a significant accuracy drop compared to the corresponding 16-bit configuration. This is due to quantization levels of weights and activations being insufficient. In this case, re-training recovers the accuracy drop entirely, matching the all-SRAM 8-bit baseline. This suggests that the errors due to non-idealities are low at 8-bit precision. However, after *freeze-**3***, the accuracy is quite low due to quantization.

Fig. 4(b) shows the improvements in energy and ANL for ResNet-18. For CIFAR-10, the *freeze-**11*** configuration for 128×128 16-bit crossbar achieves improvements of 1.58× and 4.48× in energy and ANL, respectively, while the *freeze-**15*** configuration for 64×64 16-bit crossbar achieves corresponding improvements of 1.08× and 2.5×, with negligible drop in accuracy. On the other hand, for CIFAR-100 the *freeze-**7*** configuration attains 1.51× (1.03×) and 3.77× (1.28×) improvements in energy and ANL, respectively, for both crossbar sizes with 16-bit precision. For 8-bit precision, the *freeze-**3*** configuration offers 1.73× (1.23×) and 3.15× (1.2×) improvements in energy and ANL, respectively, for 128×128 (64×64) crossbar sizes, for both datasets.

Overall, we observe that reducing the crossbar size, reduces the effect of non-idealities. This leads to a lower accuracy drop but suffers in terms of energy and ANL benefits. On the other hand, reducing the bit precision from also reduces the errors. However, the lower bit-precision limits the training ability of the SRAM block to recover the errors due to the RRAM block.

V. Conclusion

In this paper, we proposed a hybrid RRAM-SRAM architecture for efficient deep learning addressing the accuracy degradation faced by all-RRAM systems while achieving energy efficiency compared to an all-SRAM system. We analyze different partitioning configurations for various crossbar sizes and bit-precisions. We show that our proposed system coupled with a judicious selection of partitioning configuration based on area, latency, writability, and write endurance (hence, trainability) associated with each technology, can offer promising benefits for resource-constrained edge devices.

Acknowledgment

This work was supported in part by the National Science Foundation, Vannevar Bush Faculty Fellowship, and by the Center for Brain-Inspired Computing (C-BRIC), one of six centers in JUMP, funded by Semiconductor Research Corporation (SRC) and DARPA.

References

[1] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," ISCA, 2017.
[2] J. Fowers *et al.*, "A configurable cloud-scale dnn processor for real-time ai," ISCA, 2018.
[3] X. Xu *et al.*, "Scaling for edge inference of deep neural networks," *Nature Electronics*, 2018.
[4] H.-S. P. Wong *et al.*, "Metal–oxide rram," *Proceedings of IEEE*, 2012.
[5] A. Ankit *et al.*, "Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference," ASPLOS, 2019. https://github.com/Aayush-Ankit/puma-simulator/.
[6] I. Chakraborty *et al.*, "Resistive crossbars as approximate hardware building blocks for machine learning: Opportunities and challenges," *Proceedings of the IEEE*, 2020.
[7] A. Shafiee *et al.*, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ISCA*, 2016.
[8] I. Chakraborty *et al.*, "Geniex: A generalized approach to emulating non-ideality in memristive xbars using neural networks," DAC, 2020.
[9] I. Chakraborty *et al.*, "Technology aware training in memristive neuromorphic systems for nonideal synaptic crossbars," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2018.
[10] S. Jain and A. Raghunathan, "Cxdnn: Hardware-software compensation methods for deep neural networks on resistive crossbar systems," *ACM TECS*, 2019.
[11] E. J. Merced-Grafals *et al.*, "Repeatable, accurate, and high speed multi-level programming of memristor 1t1r arrays for power efficient analog computing applications," *Nanotechnology*, 2016.
[12] Y. Zhu *et al.*, "Statistical training for neuromorphic computing using memristor-based crossbars considering process variations and noise," in *DATE*, 2020.
[13] B. Crafton *et al.*, "Local learning in rram neural networks with sparse direct feedback alignment," ISLPED, 2019.
[14] Z. Meng *et al.*, "Digital offset for rram-based neuromorphic computing: A novel solution to conquer cycle-to-cycle variation," in *DATE*, 2021.
[15] I. Yoon *et al.*, "Transfer and online reinforcement learning in stt-mram based embedded systems for autonomous drones," in *DATE*, 2019.
[16] M. Donato *et al.*, "Memti: Optimizing on-chip nonvolatile storage for visual multitask inference at the edge," *IEEE Micro*, 2019.
[17] F. Chen and H. Li, "Emat: An efficient multi-task architecture for transfer learning using reram," ICCAD, 2018.
[18] G. Charan *et al.*, "Accurate inference with inaccurate rram devices: Statistical data, model transfer, and on-line adaptation," in *DAC*, 2020.
[19] C. Tan *et al.*, "A Survey on Deep Transfer Learning," Lecture Notes in Computer Science, 2018.
[20] K. He *et al.*, "Deep Residual Learning for Image Recognition," *CVPR*, 2016.